Chương 3: LẬP TRÌNH SCRIPT

3.1. Javascript

3.1.1. Giới thiệu Javascript

JavaScript là ngôn ngữ dưới dạng script có thể gắn với các file HTML. Nó không được biên dịch mà được trình duyệt diễn dịch. Không giống Java phải chuyển thành các mã để biên dịch, trình duyệt đọc JavaScript dưới dạng mã nguồn. Chính vì vậy Chúng ta có thể dễ dàng học JavaScript qua ví dụ bởi vì Chúng ta có thể thấy cách sử dụng JavaScript trên các trang Web.

JavaScript là ngôn ngữ dựa trên đối tượng, có nghĩa là bao gồm nhiều kiểu đối tượng, ví dụ đối tượng Math với tất cả các chức năng toán học. Tuy vậy JavaScript không là ngôn ngữ hướng đối tượng như C++ hay Java do không hỗ trợ các lớp hay tính thừa kế.

JavaScript có thể đáp ứng các sự kiện như tải hay loại bỏ các form. Khả năng này cho phép JavaScript trở thành một ngôn ngữ script động.

Giống với HTML và Java, JavaScript được thiết kế độc lập với hệ điều hành. Nó có thể chạy trên bất kỳ hệ điều hành nào có trình duyệt hỗ trợ JavaScript. Ngoài ra JavaScript giống Java ở khía cạnh an ninh: JavaScript không thể đọc và viết vào file của người dùng.

Các trình duyệt web như Nescape Navigator 2.0 trở đi có thể hiển thị những câu lệnh JavaScript được nhúng vào trang HTML. Khi trình duyệt yêu cầu một trang, server sẽ gửi đầy đủ nội dung của trang đó, bao gồm cả HTML và các câu lệnh JavaScript qua mạng tới client. Client sẽ đọc trang đó từ đầu đến cuối, hiển thị các kết quả của HTML và xử lý các câu lệnh JavaScript khi nào chúng xuất hiện.

Các câu lệnh JavaScript được nhúng trong một trang HTML có thể trả lời cho các sự kiện của người sử dụng như kích chuột, nhập vào một form và điều hướng trang. Ví dụ Chúng ta có thể kiểm tra các giá trị thông tin mà người sử dụng đưa vào mà không cần đến bất cứ một quá trình truyền trên mạng nào. Trang HTML với JavaScript được nhúng sẽ kiểm tra các giá trị được đưa vào và sẽ thông báo với người sử dụng khi giá trị đưa vào là không hợp lệ.

Mục đích của phần này là giới thiệu về ngôn ngữ lập trình JavaScript để Chúng ta có thể viết các script vào file HTML của mình.

3.1.2. Nhúng Javascript trong trang web

Chúng ta có thể nhúng JavaScript vào một file HTML theo một trong các cách sau đây:

- Sử dụng các câu lệnh và các hàm trong cặp thẻ <SCRIPT>
- Sử dụng các file nguồn JavaScript
- Sử dụng một biểu thức JavaScript làm giá trị của một thuộc tính HTML

- Sử dụng thẻ sự kiện (event handlers) trong một thẻ HTML nào đó

Trong đó, sử dụng cặp thẻ <SCRIPT>...</SCRIPT> và nhúng một file nguồn JavaScript là được sử dụng nhiều hơn cả.

Sử dụng thẻ SCRIPT

Script được đưa vào file HTML bằng cách sử dụng cặp thẻ <SCRIPT> và <\SCRIPT>. Các thẻ <SCRIPT> có thể xuất hiện trong phần <HEAD> hay <BODY> của file HTML. Nếu đặt trong phần <HEAD>, nó sẽ được tải và sẵn sàng trước khi phần còn lại của văn bản được tải.

Thuộc tính duy nhất được định nghĩa hiện thời cho thẻ <SCRIPT> là "LANGUAGE=" dùng để xác định ngôn ngữ script được sử dụng. Có hai giá trị được định nghĩa là "JavaScript" và "VBScript". Với chương trình viết bằng JavaScript Chúng ta sử dụng cú pháp sau :

<SCRIPT LANGUAGE="JavaScript">

// INSERT ALL JavaScript HERE

</SCRIPT>

Điểm khác nhau giữa cú pháp viết các ghi chú giữa HTML và JavaScript là cho phép Chúng ta ẩn các mã JavaScript trong các ghi chú của file HTML, để các trình duyệt cũ không hỗ trợ cho JavaScript có thể đọc được nó như trong ví dụ sau đây:

<SCRIPT LANGUAGE="JavaScript">

<!-- From here the JavaScript code hidden

// INSERT ALL JavaScript HERE

// This is where the hidden ends -->

</SCRIPT>

Dòng cuối cùng của script cần có dấu // để trình duyệt không diễn dịch dòng này dưới dạng mã JavaScript. Các ví dụ trong chương này không chứa đặc điểm ẩn của JavaScript để mã có thể dễ hiểu hơn.

Sử dụng một file nguồn JavaScript

Thuộc tính SRC của thẻ <SCRIPT> cho phép Chúng ta chỉ rõ file nguồn JavaScript được sử dụng (dùng phương pháp này hay hơn nhúng trực tiếp một đoạn lệnh JavaScript vào trang HTML).

Cú pháp:

<SCRIPT SRC="file_name.js">

....

</SCRIPT>

Thuộc tính này rất hữu dụng cho việc chia sẻ các hàm dùng chung cho nhiều trang khác nhau. Các câu lệnh JavaScript nằm trong cặp thẻ <SCRIPT> và </SCRIPT> có chứa

thuộc tinh SRC trừ khi nó có lỗi. Ví dụ Chúng ta muốn đưa dòng lệnh sau vào giữa cặp thẻ <SCRIPT SRC="..."> và </SCRIPT>:

document.write("Không tìm thấy file JS đưa vào!");

Thuộc tính SRC có thể được định rõ bằng địa chỉ URL, các liên kết hoặc các đường dẫn tuyệt đối, ví dụ:

<SCRIPT SRC=" http://cse.com.vn ">

Các file JavaScript bên ngoài không được chứa bất kỳ thẻ HTML nào. Chúng chỉ được chứa các câu lệnh JavaScript và định nghĩa hàm.

Tên file của các hàm JavaScript bên ngoài cần có đuôi .js, và server sẽ phải ánh xạ đuôi .js đó tới kiểu MIME application/x-javascript. Đó là những gì mà server gửi trở lại phần Header của file HTML. Để ánh xạ đuôi này vào kiểu MIME, ta thêm dòng sau vào file mime.types trong đường dẫn cấu hình của server, sau đó khởi động lại server:

type=application/x-javascript

Nếu server không ánh xạ được đuôi .js tới kiểu MIME application/x-javascript , Navigator sẽ tải file JavaScript được chỉ ra trong thuộc tính SRC về không đúng cách.

Trong ví dụ sau, hàm bar có chứa xâu "left" nằm trong một cặp dấu nháy kép:

```
function bar(widthPct)
{
   document.write(" <HR ALIGN='LEFT' WIDTH="+widthPct+"%>")
}
```

3.1.3. Cách đặt biểu thức cho các thuộc tính của thẻ HTML

Chúng ta có thể dùng biểu thức JavaScript làm giá trị cho thuộc tính của thẻ HTML. Các giá trị đó được thực hiện một cách động mỗi khi trang được trình duyệt tải vào. Cú pháp như sau:

& {expression};

Trong đó expression là biểu thức JavaScript sẽ được thực hiện.

Chẳng hạn ta có thể định nghĩa một biến chứa độ rộng. Chúng ta có thể dùng biến này để xác định độ rộng của một đường kẻ ngang trên trang Web. Trong ví dụ sau đây, giá trị độ rộng được đặt là 10:

Ví dụ 3.1: tạo fiel Example.html như sau

```
<HTML>
  <HEAD>
   <SCRIPT>
   var linewidth = 10;
   </SCRIPT>
  <BODY>
```

<H2>Đăt biểu thức cho các thuộc tính của HTML</H2>

<HR width="&{linewidth};%" align = "left">

</BODY>

</HTML>

Khi mã được thực thi, thì phần tử HR sẽ dùng giá trị của biến linewidth như minh họa trong hình sau:

Hình 3.1. Kết quả chạy ví dụ 3.1

3.1.4. Dùng Javascript cho trình xử lí sự kiện

JavaScript là ngôn ngữ định hướng sự kiện, nghĩa là sẽ phản ứng trước các sự kiện xác định trước như kích chuột hay tải một văn bản. Một sự kiện có thể gây ra việc thực hiện một đoạn mã lệnh (gọi là các Chương triình xử lý sự kiện) giúp cho chương trình có thể phản ứng một cách thích hợp.

Event Handler

Một đoạn mã hay một hàm được thực hiện để phản ứng trước một sự kiện gọi là Chương trình xử lý sự kiện. Chương trình xử lý sự kiện được xác định là một thuộc tính của một thẻ HTML:

tagName eventHandler = "JavaScript Code or Function"

Ví dụ sau gọi hàm CheckAge() mỗi khi giá trị của trường văn bản thay đổi:

INPUT TYPE=TEXT NAME="AGE" onChange="CheckAge()"

Đoạn mã của Chương trình xử lý sự kiện không là một hàm; nó là các lệnh của JavaScript cách nhau bằng dấu chấm phẩy. Tuy nhiên cho mục đích viết thành các module nên viết dưới dạng các hàm.

3.1.5. Các kiểu dữ liệu

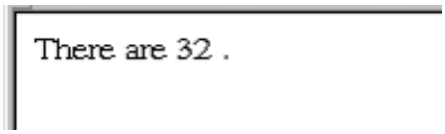
Khác với C++ hay Java, JavaScript là ngôn ngữ có tính định kiểu thấp. Điều này có nghĩa là không phải chỉ ra kiểu dữ liệu khi khai báo biến. Kiểu dữ liệu được tự động chuyển thành kiểu phù hợp khi cần thiết.

Ví dụ 3.2: file Variable.Html:

<HTML>

```
<HEAD>
<TITLE> Datatype Example </TITLE>
<SCRIPT LANGUAGE= "JavaScript">
var fruit='apples';
var numfruit=12;
numfruit = numfruit + 20;
var temp ="There are " + numfruit + " " + ".";
document.write(temp);
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Các trình duyệt hỗ trợ JavaScript sẽ xử lý chính xác ví dụ trên và đưa ra kết quả dưới đây:



Trình diễn dịch JavaScript sẽ xem biến numfruit có kiểu nguyên khi cộng với 20 và có kiểu chuỗi khi kết hợp với biến temp.

Trong JavaScript, có bốn kiểu dữ liệu sau đây: kiểu số nguyên, kiểu dấu phẩy động, kiểu logic và kiểu chuỗi.

Kiểu nguyên (Interger)

Số nguyên có thể được biểu diễn theo ba cách:

- Hệ cơ số 10 (hệ thập phân) - có thể biểu diễn số nguyên theo cơ số 10, chú ý rằng chữ số đầu tiên phải khác 0.
- Hệ cơ số 8 (hệ bát phân) - số nguyên có thể biểu diễn dưới dạng bát phân với chữ số đầu tiên là số 0.
- Hệ cơ số 16 (hệ thập lục phân) - số nguyên có thể biểu diễn dưới dạng thập lục phân với hai chữ số đầu tiên là 0x.

Kiểu dấu phẩy động (Floating Point)

Một literal có kiểu dấu phẩy động có 4 thành phần sau:

- Phần nguyên thập phân.
- Dấu chấm thập phân (.).
- Phần dư.
- Phần mũ.

Để phân biệt kiểu dấu phẩy động với kiểu số nguyên, phải có ít nhất một chữ số theo sau dấu chấm hay E. Ví dụ:

9.87

-0.85E4

9.87E14

.98E-3

Kiểu logic (Boolean)

Kiểu logic được sử dụng để chỉ hai điều kiện : đúng hoặc sai. Miền giá trị của kiểu này chỉ có hai giá trị:

- *true.*
- *false*.

Kiểu chuỗi (String)

Một literal kiểu chuỗi được biểu diễn bởi không hay nhiều ký tự được đặt trong cặp dấu " ... " hay '... '. Ví dụ:

“The dog ran up the tree”

The dog barked

“100”

Để biểu diễn dấu nháy kép ("), trong chuỗi sử dụng (\"), ví dụ:

document.write(“ \”This text inside quotes.\” ”);

3.1.6. Các toán tử

Toán tử được sử dụng để thực hiện một phép toán nào đó trên dữ liệu. Một toán tử có thể trả lại một giá trị kiểu số, kiểu chuỗi hay kiểu logic. Các toán tử trong JavaScript có thể được nhóm thành các loại sau đây: *gán, so sánh, số học, chuỗi, logic* và *logic bitwise.*

Gán

Toán tử gán là dấu bằng (=) nhằm thực hiện việc gán giá trị của toán hạng bên phải cho toán hạng bên trái. Bên cạnh đó JavaScript còn hỗ trợ một số kiểu toán tử gán rút gọn.

Kiểu gán thông thường	***Kiểu gán rút gọn***
x = x + y	x + = y
x = x – y	x - = y
x = x * y	x * = y
x = x / y	x / = y
x = x % y	x % = y

So sánh

Người ta sử dụng toán tử so sánh để so sánh hai toán hạng và trả lại giá trị đúng hay sai phụ thuộc vào kết quả so sánh. Sau đây là một số toán tử so sánh trong JavaScript:

==	Trả lại giá trị đúng nếu toán hạng bên trái bằng toán hạng bên phải
!=	Trả lại giá trị đúng nếu toán hạng bên trái khác toán hạng bên phải
>	Trả lại giá trị đúng nếu toán hạng bên trái lớn hơn toán hạng bên phải
>=	Trả lại giá trị đúng nếu toán hạng bên trái lớn hơn hoặc bằng toán hạng bên phải
<	Trả lại giá trị đúng nếu toán hạng bên trái nhỏ hơn toán hạng bên phải
<=	Trả lại giá trị đúng nếu toán hạng bên trái nhỏ hơn hoặc bằng toán hạng bên phải

Số học

Bên cạnh các toán tử cộng (+), trừ (-), nhân (*), chia (/) thông thường, JavaScript còn hỗ trợ các toán tử sau đây:

var1% var2	Toán tử phần dư, trả lại phần dư khi chia var1 cho var2
-	Toán tử phủ định, có giá trị phủ định toán hạng
var++	Toán tử này tăng var lên 1 (có thể biểu diễn là ++var)
var--	Toán tử này giảm var đi 1 (có thể biểu diễn là --var)

Chuỗi

Khi được sử dụng với chuỗi, toán tử + được coi là kết hợp hai chuỗi.

ví dụ:

"abc" + "xyz" được "abcxyz"

Logic

JavaScript hỗ trợ các toán tử logic sau đây:

expr1 && expr2	Là toán tử logic AND, trả lại giá trị đúng nếu cả expr1 và expr2 cùng đúng.
expr1 || expr2	Là toán tử logic OR, trả lại giá trị đúng nếu ít nhất một trong hai expr1 và expr2 đúng.
! expr	Là toán tử logic NOT phủ định giá trị của expr.

Bitwise

Với các toán tử thao tác trên bit, đầu tiên giá trị được chuyển dưới dạng số nguyên 32 bit, sau đó lần lượt thực hiện các phép toán trên từng bit.

& Toán tử bitwise AND, trả lại giá trị 1 nếu cả hai bit cùng là 1.

| Toán tử bitwise OR, trả lại giá trị 1 nếu một trong hai bit là 1.

^ Toán tử bitwise XOR, trả lại giá trị 1 nếu hai bit có giá trị khác nhau

Ngoài ra còn có một số toán tử dịch chuyển bitwise. Giá trị được chuyển thành số nguyên 32 bit trước khi dịch chuyển. Sau khi dịch chuyển, giá trị lại được chuyển thành kiểu của toán hạng bên trái. Sau đây là các toán tử dịch chuyển:

<< Toán tử dịch trái. Dịch chuyển toán hạng trái sang trái một số lượng bit bằng toán hạng phải. Các bit bị chuyển sang trái bị mất và 0 thay vào phía bên phải. Ví dụ: 4<<2 trở thành 16 (số nhị phân 100 trở thành số nhị phân 10000)

>> Toán tử dịch phải. Dịch chuyển toán hạng trái sang phải một số lượng bit bằng toán hạng phải. Các bit bị chuyển sang phải bị mất và dấu của toán hạng bên trái được giữ nguyên. Ví dụ: 16>>2 trở thành 4 (số nhị phân 10000 trở thành số nhị phân 100)

>>> Toán tử dịch phải có chèn 0. Dịch chuyển toán hạng trái sang phải một số lượng bit bằng toán hạng phải. Bit dấu được dịch chuyển từ trái (giống >>). Những bit được dịch sang phải bị xoá đi. Ví dụ: -8>>>2 trở thành 1073741822 (bởi các bit dấu đã trở thành một phần của số). Tất nhiên với số dương kết quả của toán tử >> và >>> là giống nhau.

Có một số toán tử dịch chuyển bitwise rút gọn:

Kiểu bitwise thông thường	*Kiểu bitwise rút gọn*
x = x << y	x << = y
x = x >> y	x - >> y
x = x >>> y	x >>> = y
x = x & y	x & = y
x = x ^ y	x ^ = y
x = x | y	x | = y

3.1.7. Các biểu thức

Tập hợp các literal, biến và các toán tử nhằm đánh giá một giá trị nào đó được gọi là một biểu thức (expression). Về cơ bản có ba kiểu biểu thức trong JavaScript:

- **Số học**: Nhằm để lượng giá giá trị số. Ví dụ *(3+4)+(84.5/3)* được đánh giá bằng *197.1666666667*.
- **Chuỗi**: Nhằm để đánh giá chuỗi. Ví dụ *"The dog barked" + barktone + "!"* là *The dog barked ferociously!*.
- **Logic**: Nhằm đánh giá giá trị logic. Ví dụ *temp>32* có thể nhận giá trị sai. JavaScript cũng hỗ trợ biểu thức điều kiện, cú pháp như sau:

  (condition) ? valTrue : valFalse

Nếu điều kiện *condition* được đánh giá là đúng, biểu thức nhận giá trị *valTrue,* ngược lại nhận giá trị *valFalse*. Ví dụ: *state = (temp>32) ? "liquid" : "solid"*

Trong ví dụ này biến state được gán giá trị *"liquid"* nếu giá trị của biến *temp* lớn hơn 32; trong trường hợp ngược lại nó nhận giá trị *"solid"*.

3.1.8. Khai báo biến, mảng

Tên biến trong JavaScript phải bắt đầu bằng chữ hay dấu gạch dưới. Các chữ số không được sử dụng để mở đầu tên một biến nhưng có thể sử dụng sau ký tự đầu tiên.

Phạm vi của biến có thể là một trong hai kiểu sau:

- **Biến toàn cục:** Có thể được truy cập từ bất kỳ đâu trong ứng dụng.

được khai báo như sau :

```
x = 0;
```

- **Biến cục bộ:** Chỉ được truy cập trong phạm vi chương trình mà nó khai báo. Biến cục bộ được khai báo trong một hàm với từ khoá var như sau:

```
var x = 0;
```

Biến toàn cục có thể sử dụng từ khoá var, tuy nhiên điều này không thực sự cần thiết.

Mảng (Array)

Mặc dù JavaScript không hỗ trợ cấu trúc dữ liệu mảng nhưng Netscape tạo ra phương thức cho phép Chúng ta tự tạo ra các hàm khởi tạo mảng như sau:

```
function InitArray(NumElements){
   this.length = numElements;
   for (var x=1; x<=numElements; x++){
      this[x]=0
   }
   return this;
}
```

Nó tạo ra một mảng với kích thước xác định trước và điền các giá trị 0. Chú ý rằng thành phần đầu tiên trong mảng là độ dài mảng và không được sử dụng.

Để tạo ra một mảng, khai báo như sau:

myArray = new InitArray (10)

Nó tạo ra các thành phần từ myArray[1] đến myArray[10] với giá trị là 0. Giá trị các thành phần trong mảng có thể được thay đổi như sau:

```
myArray[1] = "Nghệ An"
myArray[2] = "Lào"
```

Sau đây là ví dụ đầy đủ:

```
<HTML> <HEAD>
<TITLE> Array Exemple </TITLE>
```

```
<SCRIPT LANGUAGE= "JavaScript">
function InitArray(numElements) {
    this.length = numElements;
    for (var x=1; x<=numElements; x++){
        this[x]=0
    }
    return this;
}
myArray = new InitArray(10);
myArray[1] = "Nghệ An";
myArray[2] = "Hà Nội";
document.write(myArray[1] + "<BR>");
document.write(myArray[2] + "<BR>");
</SCRIPT>
</HEAD>
<BODY> </BODY>
</HTML>
```

3.1.9. Cách lệnh điều kiện

Câu lệnh điều kiện cho phép chương trình ra quyết định và thực hiện công việc nào đấy dựa trên kết quả của quyết định. Trong JavaScript, câu lệnh điều kiện là if...else

Câu lệnh này cho phép Chúng ta kiểm tra điều kiện và thực hiện một nhóm lệnh nào đấy dựa trên kết quả của điều kiện vừa kiểm tra. Nhóm lệnh sau else không bắt buộc phải có, nó cho phép chỉ ra nhóm lệnh phải thực hiện nếu điều kiện là sai.

Cú pháp:

```
if ( <điều kiện> )
{
        <Công việc 1>
}
else
{
        <Công việc 2>
}
```

Ví dụ 3.3:

```
if (x==10){
document.write("x bằng 10, đặt lại x bằng 0.");
x = 0;
```

}

else

document.write(“x không bằng 10.”);

3.1.10. Các lệnh lặp

Câu lệnh lặp thể hiện việc lặp đi lặp lại một đoạn mã cho đến khi biểu thức điều kiện được đánh giá là đúng. JavaScipt cung cấp hai kiểu câu lệnh lặp:

- **for loop**
- **while loop**

Vòng lặp for

Vòng lặp for thiết lập một biểu thức khởi đầu - initExpr, sau đó lặp một đoạn mã cho đến khi biểu thức <điều kiện> được đánh giá là đúng. Sau khi kết thúc mỗi vòng lặp, biểu thức incrExpr được đánh giá lại.

Cú pháp:

for (<Biểu thức khởi tạo>; <điều kiện> ; <Biểu thức tăng giảm>){

<Công việc>

}

Ví dụ 3.4:

<HTML> <HEAD>

<TITLE>For loop Example </TITLE>

<SCRIPT LANGUAGE= "JavaScript">

for (x=1; x<=10 ; x++) {

*y=x*25;*

*document.write("x ="+ x +";y= "+ y + "
");*

}

</SCRIPT>

</HEAD>

<BODY></BODY>

</HTML>

While

Vòng lặp while lặp khối lệnh chừng nào <điều kiện> còn được đánh giá là đúng

Cú pháp:

while (<điều kiện>)

{

<Công việc>

}

Ví dụ 3.5:

```
x=1;
while (x<=10){
    y=x*25;
    document.write("x="+x +"; y = "+ y + "<BR>");
    x++;
}
```

Kết quả của ví dụ này giống như ví dụ trước.

Break

Câu lệnh break dùng để kết thúc việc thực hiện của vòng lặp for hay while. Chương trình được tiếp tục thực hiện tại câu lệnh ngay sau chỗ kết thúc của vòng lặp.

Cú pháp :

break;

Đoạn mã sau lặp cho đến khi x lớn hơn hoặc bằng 100. Tuy nhiên nếu giá trị x đưa vào vòng lặp nhỏ hơn 50, vòng lặp sẽ kết thúc

Ví dụ 3.6:

```
while (x<100)
{
    if (x<50) break;
    x++;
}
```

Continue

Lệnh continue giống lệnh break nhưng khác ở chỗ việc lặp được kết thúc và bắt đầu từ đầu vòng lặp. Đối với vòng lặp while, lệnh continue điều khiển quay lại <điều kiện>; với for, lệnh continue điều khiển quay lại incrExpr.

Cú pháp :

Continue;

Ví dụ 3.7: Đoạn mã sau tăng x từ 0 lên 5, nhảy lên 8 và tiếp tục tăng lên 10

```
x=0;
while (x<=10)
{
    document.write("Giá trị của x là:"+ x+"<BR>");
    if (x=5)
    {
        x=8;
        continue;
    }
```

```
    x++;
}
```

For...In

Câu lệnh này được sử dụng để lặp tất cả các thuộc tính (properties) của một đối tượng. Tên biến có thể là một giá trị bất kỳ, chỉ cần thiết khi Chúng ta sử dụng các thuộc tính trong vòng lặp. Ví dụ sau sẽ minh hoạ điều này

Cú pháp:

```
for (<variable> in <object>)
{
    <Công việc>
}
```

Ví dụ 3.8:

Ví dụ sau sẽ lấy ra tất cả các thuộc tính của đối tượng Window và in ra tên của mỗi thuộc tính. Kết quả được minh hoạ trên hình 5.2.

```
<HTML>
<HEAD>
<TITLE>For in Example </TITLE>
<SCRIPT LANGUAGE= "JavaScript">
document.write("The properties of the Window object are: <BR>");
for (var x in window)
    document.write("   "+ x + ", ");
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

3.1.11. Hàm (function)

JavaScript cũg cho phép sử dụng các hàm. Mặc dù không nhất thiết phải có, song các hàm có thể có một hay nhiều tham số truyền vào và một giá trị trả về. Bởi vì JavaScript là ngôn ngữ có tính định kiểu thấp nên không cần định nghĩa kiểu tham số và giá trị trả về của hàm. Hàm có thể là thuộc tính của một đối tượng, trong trường hợp này nó được xem như là phương thức của đối tượng đó.

Lệnh function được sử dụng để tạo ra hàm trong JavaScript.

Cú pháp:

```
function fnName([param1],[param2],...,[paramN])
{
```

//function statement

}

Ví dụ 3.9:

function person(first_name, last_name, age, sex)

{

this.first_name=first_name;

this.last_name=last_name;

this.age=age;

this.sex=sex;

this.printStats=printStats;

}

Các hàm có sẵn

JavaScript có một số hàm có sẵn, gắn trực tiếp vào chính ngôn ngữ và không nằm trong một đối tượng nào:

- eval
- parseInt
- parseFloat

Eval

Hàm này được sử dụng để đánh giá các biểu thức hay lệnh. Biểu thức, lệnh hay các đối tượng của thuộc tính đều có thể được đánh giá. Đặc biệt hết sức hữu ích khi đánh giá các biểu thức do người dùng đưa vào (ngược lại có thể đánh giá trực tiếp).

Cú pháp:

returnval=eval (bất kỳ biểu thức hay lệnh hợp lệ trong Java)

Parseint

Hàm này chuyển một chuỗi số thành số nguyên với cơ số là tham số thứ hai (tham số này không bắt buộc). Hàm này thường được sử dụng để chuyển các số nguyên sang cơ số 10 và đảm bảo rằng các dữ liệu được nhập dưới dạng ký tự được chuyển thành số trước khi tính toán. Trong trường hợp dữ liệu vào không hợp lệ, hàm parseInt sẽ đọc và chuyển dạng chuỗi đến vị trí nó tìm thấy ký tự không phải là số. Ngoài ra hàm này còn cắt dấu phẩy động.

Cú pháp:

parseInt (string, [, radix])

Parsefloat

Hàm này giống hàm parseInt nhưng nó chuyển chuỗi thành số biểu diễn dưới dạng dấu phẩy động.

Cú pháp:

parseFloat (string)

3.1.12. Các đối tượng trong Javascript

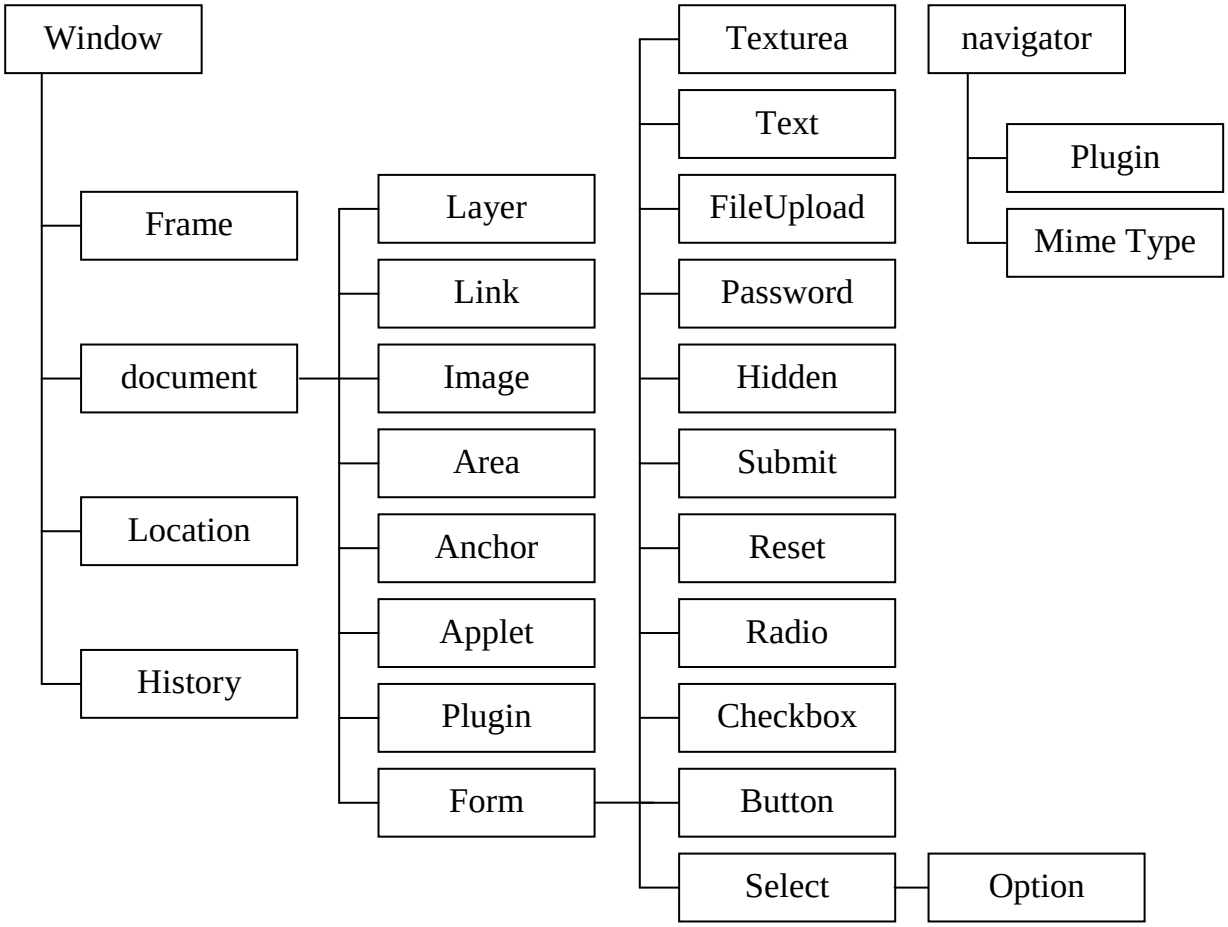
Như đã nói JavaScript là ngôn ngữ lập trình dựa trên đối tượng, nhưng không hướng đối tượng bởi vì nó không hỗ trợ các lớp cũng như tính thừa kế. Phần này nói về các đối tượng trong JavaScript và hình 6.1 chỉ ra sơ đồ phân cấp các đối tượng.

Trong sơ đồ phân cấp các đối tượng của JavaScript, các đối tượng con thực sự là các thuộc tính của các đối tượng bố mẹ. Trong ví dụ về chương trình xử lý sự kiện trước đây form tên PHIEU_DIEU_TRA là thuộc tính của đối tượng document và trường text AGE là thuộc tính của form PHIEU_DIEU_TRA. Để tham chiếu đến giá trị của AGE, Chúng ta phải sử dụng: **document.PHIEU_DIEU_TRA.AGE.value**

Các đối tượng có thuộc tính (properties), phương thức (methods), và các chương trình xử lý sự kiện (event handlers) gắn với chúng. Ví dụ đối tượng document có thuộc tính title phản ánh nội dung của thẻ <TITLE> của document. Bên cạnh đó Chúng ta thấy phương thức document.write được sử dụng trong nhiều ví dụ để đưa văn bản kết quả ra document.

Đối tượng cũng có thể có các chương trình xử lý sự kiện. Ví dụ đối tượng link có hai chương trình xử lý sự kiện là onClick và onMouseOver. onClick được gọi khi có đối tượng link được kích chuột vào, onMouseOver được gọi khi con trỏ chuột di chuyển qua link.

Khi Chúng ta tải một document xuống Navigator, nó sẽ tạo ra một số đối tượng cùng với những giá trị các thuộc tính của chúng dựa trên file HTML của document đó và một vài thông tin cần thiết khác. Những đối tượng này tồn tại một cách có cấp bậc và phản ánh chính cấu trúc của file HTML đó.



Hình 3.3. Các đối tượng của JavaScript

1) Đối tượng navigator

Đối tượng này được sử dụng để đạt được các thông tin về trình duyệt như số phiên bản. Đối tượng này không có phương thức hay chương trình xử lý sự kiện.

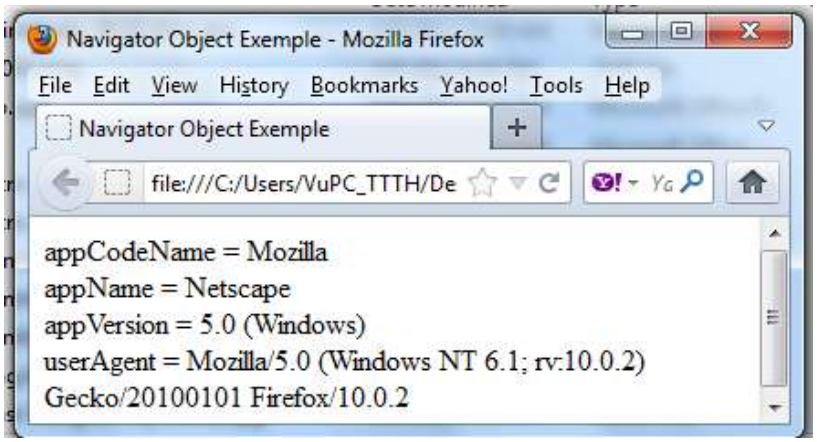
Các thuộc tính

appCodeName	Xác định tên mã nội tại của trình duyệt (Atlas).
AppName	Xác định tên trình duyệt.
AppVersion	Xác định thông tin về phiên bản của đối tượng navigator.
userAgent	Xác định header của user - agent.

Ví dụ 3.10: Hiển thị các thuộc tính của đối tượng navigator:

```
<HTML>
<HEAD>
<TITLE> Navigator Object Exemple </TITLE>
<SCRIPT LANGUAGE= "JavaScript">
document.write("appCodeName = "+navigator.appCodeName + "<BR>");
```

```
document.write("appName = "+navigator.appName + "<BR>");
document.write("appVersion = "+navigator.appVersion + "<BR>");
document.write("userAgent = "+navigator.userAgent + "<BR>");
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```



Hình 3.3: Minh hoạ cho đối tượng Navigator

2) Đối tượng window

Đối tượng window như đã nói ở trên là đối tượng ở mức cao nhất. Các đối tượng document, frame, vị trí đều là thuộc tính của đối tượng window.

Các thuộc tính

- defaultStatus - Thông báo ngầm định hiển thị lên trên thanh trạng thái của cửa sổ
- Frames - Mảng xác định tất cả các frame trong cửa sổ.
- Length - Số lượng các frame trong cửa sổ cha mẹ.
- Name - Tên của cửa sổ hiện thời.
- Parent - Đối tượng cửa sổ cha mẹ
- Self - Cửa sổ hiện thời.
- Status - Được sử dụng cho thông báo tạm thời hiển thị lên trên thanh thạng thái cửa sổ. Được sử dụng để lấy hay đặt lại thông báo trạng thái và ghi đè lên defaultStatus.
- Top - Cửa sổ ở trên cùng.
- Window - Cửa sổ hiện thời.

Các phương thức

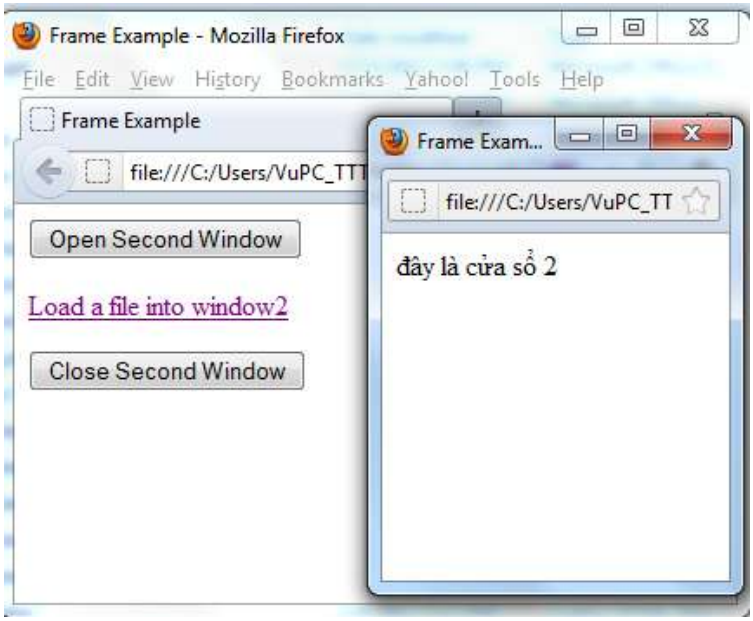
- alert ("message") -Hiển thị hộp hội thoại với chuỗi "message" và nút OK.

- clearTimeout(timeoutID) -Xóa timeout do SetTimeout đặt. SetTimeout trả lại timeoutID
- windowReference.close -Đóng cửa sổ windowReference.
- confirm("message") -Hiển thị hộp hội thoại với chuỗi "message", nút OK và nút Cancel. Trả lại giá trị True cho OK và False cho Cancel.
- [windowVar =][window]. open("URL", "windowName", ["windowFeatures"]) - Mở cửa sổ mới.
- prompt ("message" [,"defaultInput"]) - Mở một hộp hội thoại để nhận dữ liệu vào trường text.
- TimeoutID = setTimeout(expression,msec) - Đánh giá biểu thức expresion sau thời gian msec.

Ví dụ 3.11: Sử dụng tên cửa sổ khi gọi tới nó như là đích của một form submit hoặc trong một Hipertext link (thuộc tính TARGET của thẻ FORM và A).

Trong ví dụ tạo ra một tới cửa sổ thứ hai, như nút thứ nhất để mở một cửa sổ rỗng, sau đó một liên kết sẽ tải file doc2.html xuống cửa sổ mới đó rồi một nút khác dùng để đóng của sổ thứ hai lại, ví dụ này lưa vào file *window.html*:

```
<HTML>
<HEAD>
<TITLE>Frame Example </TITLE>
</HEAD>
<BODY>
<FORM>
    <INPUT TYPE="button" VALUE="Open Second Window"
onClick="msgWindow=window.open('','window2','resizable=no,width=200,height=2
00')">
    <P>
    <A HREF="doc2.html" TARGET="window2">
    Load a file into window2 </A>
    </P>
    <INPUT TYPE="button" VALUE="Close Second Window"
      onClick="msgWindow.close()">
</FORM>
</BODY>
</HTML>
```



Hình 3.4*:* Minh hoạ cho đối tượng cửa sổ

Các chương trình xử lý sự kiện

- onLoad - Xuất hiện khi cửa sổ kết thúc việc tải.
- onUnLoad - Xuất hiện khi cửa sổ được loại bỏ.

3) Đối tượng Location

Các thuộc tính của đối tượng location duy trì các thông tin về URL của document hiện thời. Đối tượng này hoàn toàn không có các phương thức và chương trình xử lý sự kiện đi kèm. Ví dụ 3.12: http:// www.abc.com/ chap1/page2.html#topic3

Các thuộc tính

- hash - Tên anchor của vị trí hiện thời (ví dụ topic3).
- Host - Phần hostname:port của URL (ví dụ www.abc.com). Chú ý rằng đây thường là cổng ngầm định và ít khi được chỉ ra.
- Hostname - Tên của host và domain (ví dụ www.abc.com).
- href - Toàn bộ URL cho document hiện tại.
- Pathname - Phần đường dẫn của URL (ví dụ /chap1/page2.html).
- Port - Cổng truyền thông được sử dụng cho máy tính host, thường là cổng ngầm định.
- Protocol - Giao thức được sử dụng (cùng với dấu hai chấm) (ví dụ http:).
- Search - Câu truy vấn tìm kiếm có thể ở cuối URL cho các script CGI.

4) Đối tượng Frame

Một cửa số có thể có một vài frame. Các frame có thể cuộn một cách độc lập với nhau và mỗi frame có URL riêng. frame không có các chương trình xử lý sự kiện. Sự kiện onLoad và onUnLoad là của đối tượng window.

Các thuộc tính

- frames - Mảng tất cả các frame trong cửa sổ.
- Name - Thuộc tính NAME của thẻ <FRAME>
- Length - Số lượng các frame con trong một frame.
- Parent - Cửa sổ hay frame chứa nhóm frame hiện thời.
- self - frame hiện thời.
- Window - frame hiện thời.

Các phương thức

- clearTimeout (timeoutID) - Xoá timeout do setTimeout lập. SetTimeout trả lại timeoutID.
- TimeoutID = setTimeout (expression,msec) - Đánh giá expression sau khi hết thời gian msec.

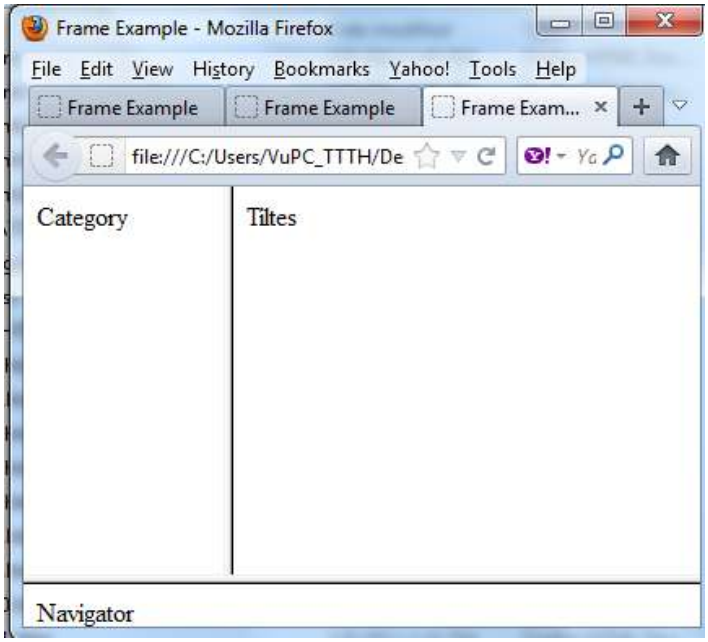
Sử dụng frame

Để tạo một frame, ta sử dụng thẻ **FRAMESET**. Mục đích của thẻ này là định nghĩa một tập các frame trong một trang.

Ví dụ 3.13*:* tạo frame

```
<HTML>
<HEAD>
<TITLE>Frame Example </TITLE>
<FRAMESET ROWS="90%,10%">
<FRAMESET COLS="30%,70%">
        <FRAME SRC=CATEGORY.HTM NAME="ListFrame">
        <FRAME SRC=TITLES.HTM NAME="contentFrame">
</FRAMESET >
<FRAME SRC=NAVIGATOR.HTM NAME="navigateFrame">
</FRAMESET >
</HEAD>
<BODY> </BODY>
</HTML>
```

Sơ đồ sau hiển thị cấu trúc của các frame: Cả 3 frame đều trên cùng một cửa sổ cha, mặc dù 2 trong số các frame đó nằm trong một frameset khác.



Hình 3.5*:* Kết quả việc tạo frame trong

5) Đối tượng document

Đối tượng này chứa các thông tin về document hiện thời và cung cấp các phương thức để đưa thông tin ra màn hình. Đối tượng document được tạo ra bằng cặp thẻ <BODY> và </BODY>. Một số các thuộc tính gắn với thẻ <BODY>.

Các đối tượng anchor, forms, history, links là thuộc tính của đối tượng document. Không có các chương trình xử lý sự kiện cho các frame. Sự kiện onLoad và onUnLoad là cho đối tượng window.

Các thuộc tính

- alinkColor - Giống như thuộc tính ALINK.
- anchor - Mảng tất cả các anchor trong document.
- bgColor - Giống thuộc tính BGCOLOR.
- cookie - Sử dụng để xác định cookie.
- fgColor - Giống thuộc tính TEXT.
- forms - Mảng tất cả các form trong document.
- lastModified - Ngày cuối cùng văn bản được sửa.
- linkColor - Giống thuộc tính LINK.
- links - Mảng tất cả các link trong document.
- location - URL đầy đủ của văn bản.
- referrer - URL của văn bản gọi nó.
- title - Nội dung của thẻ <TITLE>.
- vlinkColor - Giống thuộc tính VLINK.

Các phương thức

- document.clear - Xoá document hiện thời.

- document.close - Đóng dòng dữ liệu vào và đưa toàn bộ dữ liệu trong bộ đệm ra màn hình.
- document.open (["mineType"]) - Mở một stream để thu thập dữ liệu vào của các phwong thức write và writeln.
- document.write(expression1 [,expression2]...[,expressionN]) - Viết biểu thức HTML lên văn bản trông một cửa sổ xác định.
- document.writeln (expression1 [,expression2] ... [,expressionN]) - Giống phương thức trên nhưng khi hết mỗi biểu thức lại xuống dòng.

6) Đối tượng Anchors

Anchor là một đoạn văn bản trong document có thể dùng làm đích cho một siêu liên kết. Nó được xác định bằng cặp thẻ <A> và </A>. Đối tượng anchor không có thuộc tính, phương thức cũng như chương trình xử lý sự kiện. Mảng anchor tham chiếu đến mỗi anchor có tên trong document. Mỗi anchor được tham chiếu bằng cách:

document.anchors [index]

Mảng anchor có một thuộc tính duy nhất là length xác định số lượng các anchor trong document, nó có thể được xác định như sau:

document.anchors.length

7) Đối tượng Forms

Các form được tạo ra nhờ cặp thẻ <FORM> và </FORM>. Phần lớn các thuộc tính của đối tượng form phản ánh các thuộc tính của thẻ <FORM>. Có một vài phần tử (elements) là thuộc tính của đối tượng forms:

- button
- checkbox
- hidden
- password
- radio
- reset
- select
- submit
- text
- textarea

Các phần tử này sẽ được trình bày sau.

Nếu document chứa một vài form, chúng có thể được tham chiếu qua mảng forms. Số lượng các form có thể được xác định như sau:

document.forms.length.

Mỗi một form có thể được tham chiếu như sau:

document.forms[index]

Các thuộc tính

- action thuộc tính ACTION của thẻ FORM.
- elements Mảng chứa tất cả các thành phần trong một form (như checkbox, trường text, danh sách lựa chọn
- encoding Xâu chứa kiểu MIME được sử dụng để mã hoá nội dung của form gửi cho server.
- length Số lượng các thành phần trong một form.
- method Thuộc tính METHOD.
- target Xâu chứa tên của cửa sổ đích khi submit form

Các phương thức

- formName.submit () - Xuất dữ liệu của một form tên formName tới trang xử lý. Phương thức này mô phỏng một click vào nút submit trên form.

Các chương trình xử lý sự kiện

- onSubmit - Chương trình xử lý sự kiện này được gọi khi người sử dụng chuyển dữ liệu từ form đi.

8) Đối tượng History

Đối tượng này được sử dụng để lưu giữ các thông tin về các URL trước được người sử dụng sử dụng. Danh sách các URL được lưu trữ theo thứ tự thời gian. Đối tượng này không có chương trình xử lý sự kiện.

Các thuộc tính

- length - Số lượng các URL trong đối tượng.

Các phương thức

- history.back() - Được sử dụng để tham chiếu tới URL mới được thăm trước đây.
- history.forward() - Được sử dụng để tham chiếu tới URL kế tiếp trong danh sách. Nó sẽ không gây hiệu ứng gì nếu đã đến cuối của danh sách.
- history.go (delta | "location") - Được sử dụng để chuyển lên hay chuyển xuống delta bậc hay di chuển đến URL xác định bởi location trong danh sách. Nếu delta được sử dụng thì việc dịch chuyển lên phía trên khi delta dương và xuống phía dưới khi delta âm. nếu sử dụng location, URL gần nhất có chứa location là chuỗi con sẽ được tham chiếu.

9) Đối tượng Links

Đối tượng link là một đoạn văn bản hay một ảnh được xem là một siêu liên kết. Các thuộc tính của đối tượng link chủ yếu xử lý về URL của các siêu liên kết. Đối tượng link cũng không có phương thức nào.

Mảng link chứa danh sách tất cả các liên kết trong document. Có thể xác định số lượng các link qua

document.links.length()

Có thể tham chiếu tới một liên kết qủa

document.links [index]

Để xác định các thuộc tính của đối tượng link, có thể sử dụng URL tương tự:

http://www.abc.com/chap1/page2.html#topic3

Các thuộc tính

- hash - Tên anchor của vị trí hiện thời (ví dụ topic3).
- Host - Phần hostname:port của URL (ví dụ www.abc.com). Chú ý rằng đây thường là cổng ngầm định và ít khi được chỉ ra.
- Hostname - Tên của host và domain (ví dụ ww.abc.com).
- href - Toàn bộ URL cho document hiện tại.
- Pathname - Phần đường dẫn của URL (ví dụ /chap1/page2.html).
- port - Cổng truyền thông được sử dụng cho máy tính host, thường là cổng ngầm định.
- Protocol - Giao thức được sử dụng (cùng với dấu hai chấm) (ví dụ http:).
- Search - Câu truy vấn tìm kiếm có thể ở cuối URL cho các script CGI.
- **Target - Giống thuộc tính TARGET của <LINK>.**

Các chương trình xử lý sự kiện

- onClick - Xảy ra khi người sử dụng nhấn vào link.
- onMouseOver - Xảy ra khi con chuột di chuyển qua link.

10) Đối tượng Math

Đối tượng Math là đối tượng nội tại trong JavaScript. Các thuộc tính của đối tượng này chứa nhiều hằng số toán học, các hàm toán học, lượng giác phổ biến. Đối tượng Math không có chương trình xử lý sự kiện.

Việc tham chiếu tới ***number*** trong các phương thức có thể là số hay các biểu thức được đnáh giá là số hợp lệ.

Các thuộc tính

- E - Hằng số Euler, khoảng 2,718.
- LN2 - logarit tự nhiên của 2, khoảng 0,693.

- LN10 - logarit tự nhiên của 10, khoảng 2,302.
- LOG2E - logarit cơ số 2 của e, khoảng 1,442.
- PI - Giá trị của π, khoảng 3,14159.
- SQRT1_2 - Căn bậc 2 của 0,5, khoảng 0,707.
- SQRT2 - Căn bậc 2 của 2, khoảng 1,414.

Các phương thức

- Math.abs (*number*) - Trả lại giá trị tuyệt đối của *number*.
- Math.acos (*number*) - Trả lại giá trị arc cosine (theo radian) của *number*. Giá trị của *number* phải năng giữa -1 và 1.
- Math.asin (*number*) - Trả lại giá trị arc sine (theo radian) của *number*. Giá trị của *number* phải năng giữa -1 và 1.
- Math.atan (*number*) - Trả lại giá trị arc tan (theo radian) của *number*.
- Math.ceil (*number*) - Trả lại số nguyên nhỏ nhất lớn hơn hoặc bằng *number*.
- Math.cos (*number*) - Trả lại giá trị cosine của *number*.
- Math.exp (*number*) - Trả lại giá trị e^ *number*, với e là hằng số Euler.
- Math.floor (*number*) - Trả lại số nguyên lớn nhất nhỏ hơn hoặc bằng *number*.
- Math.log (*number*) - Trả lại logarit tự nhiên của *number*.
- Math.max (*num1*,*num2*) - Trả lại giá trị lớn nhất giữa num1 và num2
- Math.min (*num1*,*num2*) - Trả lại giá trị nhỏ nhất giữa num1 và num2.
- Math.pos (*base*,*exponent*) - Trả lại giá trị base luỹ thừa exponent.
- Math.random (*r*) - Trả lại một số ngẫu nhiên giữa 0 và 1. Phwong thức này chỉ thực hiện được trên nền tảng UNIX.
- Math.round (*number*) - Trả lại giá trị của *number* làm tròn tới số nguyên gần nhất.
- Math.sin (*number*) - Trả lại sin của *number*.
- Math.sqrt (*number*) - Trả lại căn bậc 2 của *number*.
- Math.tan (*number*) - Trả lại tag của *number*.

11) Đối tượng Date

Đối tượng Date là đối tượng có sẵn trong JavaScript. Nó cung cấp nhiều phương thức có ích để xử lý về thời gian và ngày tháng. Đối tượng Date không có thuộc tính và chương trình xử lý sự kiện.

Phần lớn các phương thức date đều có một đối tượng Date đi cùng. Các phương thức giới thiệu trong phần này sử dụng đối tượng Date dateVar, ví dụ:

dateVar = new Date ('August 16, 1996 20:45:04');

Các phương thức

- *dateVar*.getDate() - Trả lại ngày trong tháng (1-31) cho *dateVar*.
- *dateVar*.getDay() - Trả lại ngày trong tuần (0=chủ nhật,...6=thứ bảy) cho *dateVar*.
- *dateVar*.getHours() - Trả lại giờ (0-23) cho *dateVar*.
- *dateVar*.getMinutes() - Trả lại phút (0-59) cho *dateVar*.
- *dateVar*.getSeconds() - Trả lại giây (0-59) cho *dateVar*.
- *dateVar*.getTime() - Trả lại số lượng các mili giây từ 00:00:00 ngày 1/1/1970.
- *dateVar*.getTimeZoneOffset() - Trả lại độ dịch chuyểnbằng phút của giờ địa phương hiện tại so với giờ quốc tế GMT.
- *dateVar*.getYear()-Trả lại năm cho dateVar.
- Date.parse (*dateStr*) - Phân tích chuỗi *dateStr* và trả lại số lượng các mili giây tính từ 00:00:00 ngày 01/01/1970.
- *dateVar*.setDay(*day*) - Đặt ngày trong tháng là *day* cho *dateVar*.
- *dateVar*.setHours(*hours*) - Đặt giờ là *hours* cho *dateVar*.
- *dateVar*.setMinutes(*minutes*) - Đặt phút là *minutes* cho *dateVar*.
- *dateVar*.setMonths(*months*) - Đặt tháng là *months* cho *dateVar*.
- *dateVar*.setSeconds(*seconds*) - Đặt giây là s*econds* cho *dateVar*.
- *dateVar*.setTime(*value*) - Đặt thời gian là value, trong đó value biểu diễn số lượng mili giây từ 00:00:00 ngày 01/01/10970.
- *dateVar*.setYear(*years*) - Đặt năm là *years* cho *dateVar*.
- *dateVar*.toGMTString() - Trả lại chuỗi biểu diễn dateVar dưới dạng GMT.
- *dateVar*.toLocaleString()-Trả lại chuỗi biểu diễn dateVar theo khu vực thời gian hiện thời.
- Date.UTC (year, month, day [,hours] [,minutes] [,seconds]) - Trả lại số lượng mili giây từ 00:00:00 01/01/1970 GMT.

12) Đối tượng String

Đối tượng String là đối tượng được xây dựng nội tại trong JavaScript cung cấp nhiều phương thức thao tác trên chuỗi. Đối tượng này có thuộc tính duy nhất là độ dài (length) và không có chương trình xử lý sự kiện.

Các phương thức

- str.anchor (name) - Được sử dụng để tạo ra thẻ <A> (một cách động). Tham số name là thuộc tính NAME của thẻ <A>.
- str.big() - Kết quả giống như thẻ <BIG> trên chuỗi *str*.
- str.blink() - Kết quả giống như thẻ <BLINK> trên chuỗi *str*.
- str.bold() - Kết quả giống như thẻ <BOLD> trên chuỗi *str*.

- str.charAt(a) - Trả lại ký tự thứ a trong chuỗi *str*.
- str.fixed() - Kết quả giống như thẻ <TT> trên chuỗi *str*.
- str.fontcolor() - Kết quả giống như thẻ <FONTCOLOR = *color*>.
- str.fontsize(*size*) - Kết quả giống như thẻ <FONTSIZE = *size*>.
- str.index0f(srchStr [,index]) - Trả lại vị trí trong chuỗi *str* vị trí xuất hiện đầu tiên của chuỗi *srchStr*. Chuỗi *str* được tìm từ trái sang phải. Tham số *index* có thể được sử dụng để xác định vị trí bắt đầu tìm kiếm trong chuỗi.
- str.italics() - Kết quả giống như thẻ <I> trên chuỗi *str*.
- str.lastIndex0f(srchStr [,index]) - Trả lại vị trí trong chuỗi *str* vị trí xuất hiện cuối cùng của chuỗi *srchStr*. Chuỗi *str* được tìm từ phải sang trái. Tham số *index* có thể được sử dụng để xác định vị trí bắt đầu tìm kiếm trong chuỗi.
- str.link(href) - Được sử dụng để tạo ra một kết nối HTML động cho chhuỗi *str*. Tham số *href* là URL đích của liên kết.
- str.small() - Kết quả giống như thẻ <SMALL> trên chuỗi *str*.
- str.strike() - Kết quả giống như thẻ <STRIKE> trên chuỗi *str*.
- str.sub() - Tạo ra một subscript cho chuỗi *str*, giống thẻ <SUB>.
- str.substring(a,b) - Trả lại chuỗi con của *str* là các ký tự từ vị trí thứ a tới vị trí thứ b. Các ký tự được đếm từ trái sang phải bắt đầu từ 0.
- str.sup() - Tạo ra superscript cho chuỗi *str*, giống thẻ <SUP>.
- str.toLowerCase() - Đổi chuỗi *str* thành chữ thường.
- str.toUpperCase() - Đổi chuỗi *str* thành chữ hoa.

3.2. VBScript

3.2.1. Giới thiệu VBScript

VBScript là ngôn ngữ kịch bản chính để lập trình ứng dụng web phía server khi sử dụng công nghệ ASP. Các định danh trong VBScript không phân biệt chữ hoa hay chữ thường.

3.2.2. Nhúng VBScript trong trang web

Sử dụng thẻ SCRIPT

Script được đưa vào file HTML bằng cách sử dụng cặp thẻ <SCRIPT> và <\SCRIPT>. Các thẻ <SCRIPT> có thể xuất hiện trong phần <HEAD> hay <BODY> của file HTML. Nếu đặt trong phần <HEAD>, nó sẽ được tải và sẵn sàng trước khi phần còn lại của văn bản được tải.

Thuộc tính duy nhất được định nghĩa hiện thời cho thẻ <SCRIPT> là "LANGUAGE=" dùng để xác định ngôn ngữ script được sử dụng. Có hai giá trị được định

nghĩa là "JavaScript" và "VBScript". Với chương trình viết bằng VBScript Chúng ta sử dụng cú pháp sau :

<SCRIPT LANGUAGE="VBScript">

'INSERT ALL VBScript HERE

</SCRIPT>

Ngoài ra VBScript còn có thể được nhúng vào trang web theo cú pháp của asp như sau.

```
<%
   'In ra chuỗi Hello World !
   Response.Write("Hello World !")
   %>
```

3.2.3. Cách đặt biểu thức cho các thuộc tính của thẻ HTML

Chúng ta cũng có thể dùng biểu thức VBScript làm giá trị cho thuộc tính của thẻ HTML. Các giá trị đó được thực hiện một cách động mỗi khi trang được trình duyệt tải vào.

Cú pháp:

<% =expression %>

Trong đó: expression là biểu thức VBScript sẽ được thực hiện.

Ví dụ 3.14 sau sẽ đặt giá trị cho thuộc tính width bằng biểu thức Vbscript:

```
<html>
<head>
<title>Đặt biểu thức cho thuộc tính của thẻ HTML</title>
      <%
        dim a
        a=200
      %>
</head>
<body>
          <hr width = <%=a%>
</body>
</html>
```

3.2.4. Dùng VBscript cho trình xử lí sự kiện

Cũng giống như Javascript, VBScript cũng được dùng cho trình xử lý sự kiện của HTML. Sau đây là cách tạo ra một trình xử lý sự kiện sẽ gọi đến VBScript

<SCRIPT FOR="Tên đối tượng" EVENT="Sự kiện" LANGUAGE="VBScript">

// Chương trình viết bằng ngôn ngữ VBScript

</SCRIPT>

Ví dụ 3.15: sau đây tạo ra một nút gọi đến một chương trình VBScript:

<HTML>

<HEAD>

<TITLE>Test Button Events</TITLE>

</HEAD>

<BODY>

<FORM NAME="Form1">

<INPUT TYPE="Button" NAME="Button1" VALUE="Click">

<SCRIPT FOR="Button1" EVENT="onClick" LANGUAGE="VBScript">

MsgBox "Button Pressed!"

</SCRIPT>

</FORM>

</BODY>

</HTML>

3.2.5. Các kiểu dữ liệu

VBScript chỉ có một kiểu dữ liệu duy nhất là **Variant**. **Variant** là kiểu dữ liệu đặc biệt có thể chứa các loại dữ liệu khác nhau từ những kiểu dữ liệu đơn giản như kiểu số cho đến các kiểu dữ liệu phức tạp như kiểu bản ghi (record). Vì **Variant** là kiểu dữ liệu duy nhất của VBScript nên đây là cũng là kiểu dữ liệu được trả về từ các hàm/thủ tục viết bằng VBScript. Nói một cách dễ hiểu hơn là: nếu trong Pascal Chúng ta phải lưu trữ dữ liệu số trong kiểu dữ liệu **interger**, dữ liệu chuỗi trong kiểu dữ liệu **string**, thì trong VBScript Chúng ta có thể vừa lưu trữ dữ liệu số, vừa lưu trữ dữ liệu chuỗi (hay bất kì dữ liệu kiểu nào khác) trong kiểu dữ liệu **Variant**. Việc xem một biến **Variant** là số hay chuỗi tùy vào ngữ cảnh sử dụng. Ví dụ: trong biểu thức **b=a+1234** thì **a, b** được xem như là kiểu số còn trong biểu thức **b=a+"1234"**, thì **a, b** được xem như là kiểu chuỗi.

Variant Subtypes: là các dạng thông tin khác nhau mà kiểu dữ liệu **Variant** có thể lưu trữ.

Ngoài việc đơn giản là phân biệt số và xâu, một variant có thể phân biệt được thông tin số theo cách khác. Chảng hạn như Chúng ta có thể có một dữ liệu số đại diện cho Date/Time. Khi Chúng ta sử dụng nó cùng với một dữ liệu kiểu Date/Time khác thì kết quả trả về luôn được biểu diễn dưới dạng Date/Time. Tất nhiên Chúng ta có thể còn có một loạt các dữ liệu dạng số với kích thước khác nhau từ kiểu Boolean cho tới kiểu floating – point. Các dạng thông tin khác nhau đó có thể được lưu trong biến variant gọi là các kiểu con

(subtype). Phần lớn thời gian, Chúng ta chỉ cần gán dữ liệu của Chúng ta vào biến variant và biến này sẽ hoạt động theo cách xử lý dữ liệu giống như chính dữ liệu mà nó chứa.

Bảng dưới đây mô tả các kiểu dữ liệu con của variant:

Subtype	Mô tả
Empty	Variant chưa được gán giá trị ban đầu. Có giá trị 0 đối với các biến kiểu số và xâu rỗng (“”) đối với biến xâu.
Null	Variant không chứa dữ liệu
Boolean	Có giá trị là True hoặc False
Byte	Chứa số nguyên từ 0 tới 255.
Integer	Chứa số nguyên từ -32,768 tới 32,767.
Currency	-922,337,203,685,477.5808 tới 922,337,203,685,477.5807.
Long	Chứa số nguyên từ -2,147,483,648 tới 2,147,483,647.
Single	Chứa số single-precision, floating-point từ -1.402823E38 tới -1.401298E-45 đối với giá trị âm, từ 1.401298E-45 tới 3.402823E38 đối với giá trị dương
Double	Chứa số double-precision, floating-point -1.79769313486232E308 to -4.94065645841247E-324 đối với giá trị âm, từ 4.94065645841247E-324 tới 1.79769313486232E308đối với giá trị dương.
Date (Time)	Chứa một giá trị số đại diện cho ngày tính từ January 1, 100 tới December 31, 9999.

Chúng ta có thể dùng các hàm chuyển đổi kiểu dữ liệu để chuyển dữ liệu giữa các kiểu dữ liệu con với nhau. Thêm vào đó, hàm VarType cho Chúng ta biết thông tin về cách lưu trữ dữ liệu của Chúng ta trong biến Variant.

3.2.6. Các toán tử

VBScript có một tập hợp lớn các loại toán tử, chia ra thành ba loại là các toán tử số học, các toán tử so sánh và ghép nối (concatenation) và các toán tử logical.

Thứ tự ưu tiên của các toán tử

Khi có nhiều toán tử cùng xuất hiện trong một biểu thức, từng phần của biểu thức được đánh giá và xử lý theo một trình tự gọi là thứ tự ưu tiên. Chúng ta có thể dùng dấu ngoặc đơn để thay đổi thứ tự ưu tiên và bắt một phần nào đó của biểu thức phải được thực hiện trước các phần khác. Các biểu thức bên trong dấu ngoặc đơn luôn được xử lý trước những biểu thức bên ngoài. Tất nhiên, nếu biểu thức trong ngoặc chứa nhiều toán tử thì chúng cũng phải tuân theo thứ tự ưu tiên chuẩn.

Khi các biểu thức chứa nhiều loại toán tử khác nhau, các toán tử số học được xử lý trước, sau đến các toán tử so sánh rồi cuối cùng là các toán tử logic.

Các toán tử so sánh tất cả có cùng thứ tự ưu tiên, tức là chúng sẽ được xủa lý từ trái qua phải theo thứ tự xuất hiện. Các toán tử số học và logic được xử lý theo thứ tự toán tử mũ, nhân, chia, cộng, trừ...

Bảng sau đây là danh sách các toán tử của VBScript:

Toán tử	Ký hiệu
Gán	=
Phép cộng	+
Phép trừ	-
Phép nhân	*
Phép chia	/
Mũ	^
Nối các chuỗi và giá trị	&
Chia lấy phần lẻ	Mod
Phủ định	Not
Và	And
Hoặc	Or
E-Or	Xor
So sánh bằng	=
S/s lớn hơn	>
S/s lớn hơn hoặc bằng	>=
So sánh không bằng	<>
S/s nhỏ hơn	<
S/s nhỏ hơn hoặc bằng	<=

3.2.7. Các biểu thức

Tập hợp các literal, biến và các toán tử nhằm đánh giá một giá trị nào đó được gọi là một biểu thức (expression). Về cơ bản cũng giống như JavaScript VBScript cũng có ba kiểu biểu thức.

- **Số học**: Nhằm để lượng giá giá trị số. Ví dụ *(3+4)+(84.5/3)* được đánh giá bằng *197.1666666667*.
- **Chuỗi**: Nhằm để đánh giá chuỗi. Ví dụ *"The dog barked" & barktone & "!"* là *The dog barked ferociously!*. *Như vậy khác với Javascript phép toán nói 2 chuỗi dùng phép toán + thì VBScript dùng phép toán &.*
- **Logic**: Nhằm đánh giá giá trị logic. Ví dụ *temp>32* có thể nhận giá trị sai. Biểu thức lôgic được sử dụng trong các biểu thức điều kiện của các lệnh điều khiển rẽ nhánh hoặc vòng lặp.

3.2.8. Khai báo biến, mảng

Biến là một vùng chứa thông tin mà Chúng ta cần lưu trữ. Giá trị của biến có thể được thay đổi trong quá trình lập trình. Chúng ta có thể làm việc với một biến thông qua tên của nó, cũng như có thể thay đổi giá trị của biến đó. Trong VBScript, tất cả các biến đều có kiểu là variant, và nó có thể lưu trữ bất kỳ dạng dữ liệu nào.

Quy tắc đặt tên biến: Bắt đầu bằng một chữ cái, không chứa dấu (.) và độ dài không quá 255 ký tự. Chúng ta có thể khai báo biến với các từ khoá Dim, Public hoặc Private.

Để khai báo biến ta sử dụng cú pháp như sau:

Dim tên biến 1, tên biến 2, ...

Ví dụ dưới đây khai báo một biến tên name và gán cho nó một giá trị:

Dim name

name = giá trị

Chúng ta cũng có thể khai báo biến bằng cách sử dụng nó trong script của Chúng ta.

Ví dụ:

name = giá trị

Tuy vậy, cách khai báo này không được tường minh và không tốt cho ứng dụng của Chúng ta, vì sau đó trong ứng dụng của mình, Chúng ta có thể vô tình viết sai tên biến và có thể nhận được kết quả không chính xác khi chạy chương trình. Điều đó xảy ra là vì giả sử Chúng ta có một tên biến tên "name", sau đó Chúng ta gọi tới biến đó bằng một tên "nime" chẳng hạn, chương trình sẽ tự động sinh ra thêm 1 biến tên "nime". Để tránh xảy ra điều nhầm lẫn này, Chúng ta nên sử dụng

Câu lệnh Option Explicit. Khi sử dụng câu lênh này, tất cả các biến đều phải khai báo trước khi sử dụng bởi các câu lệnh với từ khoá Dim, Public hoặc

Private. Đặt câu lệnh Option Explicit trên đầu của chương trình của Chúng ta, như ví dụ sau:

Option Explicit

Dim name

name = giá trị

Cách gán giá trị cho biến:

Chúng ta có thể gán giá trị cho cho một biến như sau:

name = "Nguyễn Minh Phượng"

i = 200

Thời gian sống của biến

Khoảng thời gian biến đó tồn tại được gọi là thời gian sống của nó. Khi chúng ta khai báo một biến trong một thủ tục, biến đó chỉ được truy xuất tới trong phạm vi thủ tục đó. Khi thủ tục đó kết thúc, các biến đó cũng bị huỷ.

Những biến này được gọi là biến cục bộ. Chúng ta có thể đặt các biến cục bộ trùng tên nhau trong các thủ tục khác nhau, bởi vì mỗi biến chỉ được nhận biết bởi chính thủ tục trong đó chúng được khai báo. Nếu chúng ta khai báo một biến bên ngoài một thủ tục, tất cả các thủ tục nằm trong cùng trang đó đều có thể truy nhập tới biến đó. Thời gian sống của biến này bắt đầu từ lúc nó được khai báo và kết thúc khi trang web được đóng lại.

Biến mảng - Array:

Có những khi chúng ta muốn gán nhiều hơn 1 giá trị cho một biến, khi đó chúng ta khai báo một biến có thể chứa một dãy dữ liệu. Biến này được gọi là biến array.

Để khai báo một biến là biến array, chúng ta đặt dấu ngoặc đơn ngay sau tên biến.

Cú pháp :

Dim tên_biến[([số_chiều_biến_mảng])][, tên_biến[([số_chiều_biến_mảng)]] . . .

Thành phần	Diễn giải
tên_biến	Tên của biến; phải theo cú pháp tên biến thông thường.
số_chiều_biến_mảng	Chiều của biến mảng; lên đến 60 chiều có thể khai báo. Số_chiều_biến_mảng sử dụng theo cú pháp sau: biên_trên [**,**biên_trên] **. . .** Biên dưới của 1 mảng luôn luôn zero.

Ví dụ sau chúng ta khai báo một biến array gồm có 3 giá trị:

dim names(2)

Giá trị số trong dấu ngoặc là 2. Chỉ số của biến array bắt đầu bởi 0 cho nên biến này sẽ bao gồm 3 giá trị. Đây là một array có độ dài cố định. Chúng ta gán giá trị cho từng phần tử của array bằng cách sau:

names(0) = "Nguyễn Thanh Bình"

names(1)="Nguyễn Minh Phượng"

names(2)="Hoàng Khánh Hưng"

Tương tự như vậy ta có thể lấy giá trị của bất kỳ phần tử nào trong array mà chúng ta cần bằng cách sử dụng chỉ số tương ứng của phần tử:

eng = names(0)

Chúng ta có thể khai báo nhiều nhất tới 60 chiều cho một array. Các chiều được khai báo cách nhau bởi dấu phẩy. Ví dụ sau khai báo một array bao gồm 5 dòng và 7 cột:

dim table(4,6)

Khai báo mảng động

Khai báo Dim với 1 biến và cặp () rỗng để khai báo 1 array động. Sau khi khai báo 1 array động, dùng khai báo ReDim trong 1 procedure để khai báo số chiều và và các thành phần trong array. Nếu chúng ta khai báo 1lại (ReDim) 1 array đã khai báo số chiều rõ ràng bằng khai báo Dim, thì bị lỗi.

Cú pháp:

ReDim [Preserve] tên_biến(chiều) [, tên_biến(chiều)] . . .

Thành phần	Mô tả

Preserve	Bảo vệ data trong mảng đã tồn tại khi bạn thay đổi cỡ chiều sau
tên_biến	Tên biến
chiều	Chiều của biến mảng

Ví dụ 3.16:

Dim Names(9) ' Khai báo 1 mảng Names có 10 phần tử.

Dim Names() ' Khai báo mảng động.

Dim MyVar, MyNum ' Khai báo 2 biến.

Khai báo hằng

Để khai báo một hằng chúng ta sử dụng cú pháp như sau:

CONST tên hằng = giá trị

Ví dụ: *Const Max=100*

3.2.9. Cách lệnh điều kiện

Cấu trúc IF

Cấu trúc lệnh điều khiện if Thực hiện một nhóm lệnh tùy theo điều kiện, phụ thuộc và giá trị của 1 biểu thức.

Ta có 3 dạng cú pháp của lệnh if như sau:

Dạng 1:

If <biểu_thức_logic> then

<công_việc>

End if

Dạng này cho phép ta rẽ một nhánh ta tạm dịch ý nghĩa của dạng lệnh này như sau, Nếu biểu thức logic là đúng thì thực hiện công việc.

Dạng 2:

If <biểu_thức_logic> then

<công_việc_1>

Else

<công_việc_2>

End if

Ý nghĩa của lệnh dạng này như sau, nếu biểu thức logic là đúng thì thực hiện công việc 1, ngược lại thì thực hiện công việc 2. Như vậy theo dạng lệnh này thì một trong 2 công việc sẽ được thực hiện tùy thuộc vào giá trị của biêu thức điều kiện.

Dạng 3:

If <biểu_thức_logic1> then

<công_việc_1>

ElseIf <biểu_thức_logic2>

<công_việc_2>

Else

<công_việc_khác>

End if

Ý nghĩa của lệnh dạng này như sau, nếu biểu thức logic 1 là đúng thì thực hiện công việc 1, ngược lại thì kiểm tra biểu thức logic2 nếu đúng thì thực hiện công việc 2 ngược lại thực hiện công việc khác. Như vậy theo dạng lệnh này thì một trong 3 công việc sẽ được thực hiện tùy thuộc vào giá trị của các biêu thức điều kiện. Chúng ta sử dụng câu lệnh này khi muốn lựa chọn một trong nhiều tập hợp lệnh để thực hiện.

Ví dụ 3.17:

```
<% h=hour(now)
    If h >12 then
    Msgbox "Afternoon"
    else
    Msgbox "Morning"
End if %>
```

Select case

Là câu lệnh điều kiện rẽ nhánh trong trường hợp có nhiều hơn 2 lựa chọn cú pháp như sau:

Select Case <biểu_thức>

Case <giá_trị_1>

<công_việc_1>

...

Case <giá_trị_n>

<công_việc_n>

Case Else

<công_việc_khác>

End Select

Ý nghĩa của dạng lệnh này có thể được mô tả như sau, trong trường hợp biểu thức có giá trị bằng giá trị 1 thì thực hiện công việc 1, bằng giá trị 2 thì thực hiện công việc 2... có giá trị bằng giá trị n thì thực hiện công việc n, ngược lại thì thực hiện công việc n-1. Như vậy câu lệnh select case này cho phép ta thực hiện việc so sánh biểu thức với nhiều giá trị khá nhau tùy thuộc vào giá trị của biểu thức mà có thể thực hiện công việc tương ứng.

Ví dụ 3.18:

```
<%
```

```
h=hour(now)
Select case h
Case "1"
  msgbox "1 am"
Case "2"
  msgbox "2 am"
Case else
  msgbox "Other "
End select
%>
```

3.2.10. Các lệnh lặp

Vòng lặp xác định For

Dạng 1:

For biến_điều_khiển = giá_trị_đầu To giá_trị_cuối

<công_việc>

Next

Ta có thể mô tả ý nghĩa của lệnh như sau, bắt đầu từ biến bằng giá trị đầu đến khi biến bằng giá trị cuối hãy thực hiện lặp lại công việc, mỗi lần thực hiện khi gặp Next biến sẽ tự động tăng lên 1.

Dạng 2:

For biến_đkhiển = gtrị_đầu To gtrị_cuối Step n

<công_việc>

Next

Ta có thể mô tả ý nghĩa của lệnh như sau, bắt đầu từ biến bằng giá trị đầu đến khi biến bằng giá trị cuối hãy thực hiện lặp lại công việc, mỗi lần thực hiện khi gặp Next biến sẽ tự động tăng lên n. Biến n sau Step là số bước nhẩy, n mặc định là bằng 1, n là một số nguyên có thể là một số âm hoặc dương, trong trường hợp n là số âm thì vòng lặp sẽ thực hiện giảm dần, khi đó giá trị đầu phải lớn hơn giá trị cuối.

Ví dụ 3.19: Hiển thị ra mành hình 10 số:

```
For i = 1 to 10
      Msgbox( “i = “ &i)
Next
```

Vòng lặp For Each

Tương tự For ...Next nhưng thay vì lặp theo một số lần đã định trước, lệnh For Each...Next được dùng để lặp tương ứng với mỗi thành phần của các biến dạng collection hoặc mỗi thành phần của mảng.

Cú pháp:

For each biến In tập_đối_tượng

<công_việc>

Next

Ví dụ 3.20:

Để đọc ra tất cả các biến nhận được từ phương thức gửi get ta có thể thực hiện như sau:

For EACH item IN Request.QueryString

Response.Write item

NEXT

Vòng lặp không xác định do .. Loop

Chúng ta có thể dùng cấu trúc này để thực hiện một tập hợp lệnh khi không biết trước số lần cần thực hiện. Vòng lặp sẽ thực hiện khi điều kiện While vẫn còn được thỏa mãn. Chúng ta dùng từ khoá While để kiểm tra điều kiện trong cấu trúc Do...Loop có các dạng như sau:

Dạng 1:

Do

<công_việc>

Loop While <biểu_thức_logic>

Ở dạng lặp này công việc sẽ được thực hiện trước khi kiểm tra biểu thức logic. Vì vậy chắc chắn công việc sẽ được thực hiện 1 lần.

Dạng 2:

Do While <biểu_thức_logic>

<công_việc>

Loop

Trong dạng lặp này biểu thức lôgic sẽ được kiểm tra trước khi thực hiện công việc, nên có thể công việc sẽ không được thực thi lần nào.

Dạng 3:

Do

<công_việc>

Loop Until <biểu_thức_logic>

Trong dạng này từ khoá Until được sử dụng thay cho while, ý nghĩa là lặp lại cho đến khi biểu thức logic sau Until thỏa mãn.

Dạng 4:

Do Until <biểu_thức_logic>

<công_việc>

Loop

Dạng 5:

While <bt_logic>

<công_việc>

Wend

Dạng này có ý nghĩa tương tự như dạng do while, tức là sẽ thực hiện lặp lại công việc cho đến khi biểu thức logic sai.

Ví dụ 3.21:

Để thực hiện tìm ước số chung lớn nhất của 2 số a và b ta thực hiện như sau:

```
While a<>b
    If a >b then
        a = a –b
    Else
        b = b- a
    Endif
Wend
Uscln = a;
```

3.2.11. Hàm (function) và thủ tục Procedure

VBScript cũng cung cấp 2 loại chương trình con, giúp người lập trình có thể thực hiện việc modul hóa đó là chương trình con hàm và chương trình con thủ tục.

Chương trình con thủ tục - Subprocedure (Procedure) được khai báo như sau:

```
Sub tên_thủ _tục(tham_số_1, ..., tham_số_N)
    <thân_thủ_tục
     [Exit Sub]
    >
End Sub
```

Ví dụ 3.22, Ta có thể xây dựng chương trình con đổi nhiệt độ như sau:

```
Sub ConvertTemp()
      temp = InputBox("Nhập vào nhiệt độ ở độ F.", 1)
      MsgBox "Nhiệt độ la " & Celsius(temp) & " độ C."
End Sub
```

Chương trình con Hàm – Function được khai báo như sau:

```
Function tên_hàm(tham_số_1, ..., tham_số_N)
    <thân_hàm
     [Exit Function]
    >
```

End Function

Ví dụ 3.23, sau đây ta xây dựng hàm Celsius chuyển đổi nhiệt độ ở độ F sang độ C:

Function Celsius(fDegrees)

*Celsius = (fDegrees - 32) * 5 / 9*

End Function

Trong ví dụ này ta thấy tên hàm được coi như một biến, khi gọi hàm biến tên hàm sẽ nhận giá trị được trả về. Như chúng ta đã biết sự khác nhau giữa hàm và thủ tục là hàm thì tên hàm được coi như là một biến nhớ, nhờ đó hàm có thể tham gia vào một biểu thức bất kỳ, còn thủ tục thì không.

Để gọi hàm và thủ tục ta gội thông qua tên hàm và thủ tục đó ví dụ để gọi hàm celsius ta thực hiện như sau:

TempC = Celsius(Temp_F)

Còn để gọi thủ tục chuyển đổi ở trên ta thực hiện bằng 1 trong 2 cách sau:

Call ConvertTemp() hoặc ConvertTemp()

Dưới đây liệt kê các hàm có sẵn trong VBScript. Các hàm này được chia ra thành các loại sau:

- Các hàm về thời gian
- Các hàm chuyển đổi kiểu dữ liệu
- Các hàm định dạng dữ liệu
- Các hàm toán học
- Các hàm về dãy
- Các hàm về xâu
- Các hàm khác

Các hàm về thời gian (Date/Time Functions)

Tên hàm	Mô tả
CDate	Chuyển biểu thức có dạng date and time chuẩn sang dạng Date
Date	Trả về ngày giờ hệ thống
DateAdd	Trả về ngày được cộng thêm một khoảng thời gian
DateDiff	Trả về giá trị số là khoảng thời gian giữa hai giá trị ngày.
DatePart	Trả về phần xác định của ngày.
Day	Trả về ngày hiện tại. Giá trị từ 1 tới 31.
FormatDateTime	Trả về biểu thức đã được định dạng theo kiểu date or time
Hour	Trả về giá trị là một số chỉ giờ hiện hành trong ngày, có giá trị từ 0 tới 23.

IsDate	Trả về giá trị Boolean cho biết biểu thức có thể chuyển sang dạng ngày tháng hay không.
Minute	Trả về giá trị số là phút của giờ (có giá trị từ 0 tới 59)
Month	Cho biết tháng hiện hành (Có giá trị từ 1 tới 12)
MonthName	Trả về tên tháng
Now	Cho biết ngày giờ hiện hành của hệ thống
Second	Trả về số đại diện cho giây (Có giá trị từ 0 tới 59)
Time	Trả về giờ hệ thống
Timer	Trả về giá trị số giây tính từ 12:00 AM
Weekday	Trả về số đại diện cho ngày trong tuần (Có giá trị từ 1tới 7)
WeekdayName	Trả về tên ngày trong tuần
Year	Trả về năm hiện hành

Các hàm chuyển kiểu dữ liệu (Conversion Functions)

Tên hàm	Mô tả
Asc	Chuyển ký tự đầu tiên của xâu sang mã ANSI.
CBool	Chuyển dữ liệu kiểu variant sang kiểu subtype Boolean
CByte	Chuyển dữ liệu từ kiểu variant sang kiểu subtype Byte
CCur	Chuyển dữ liệu từ kiểu variant sang kiểu subtype Currency
CDate	Chuyển dữ liệu từ biểu thức dạng date/time sang kiểu subtype Date/Time
CDbl	Chuyển biểu thức từ kiểu variant sang kiểu subtype Double
Chr	Chuyển mã ANSI sang ký tự
CInt	Chuyển dữ liệu kiểu variant sang kiểu subtype Integer
CLng	Chuyển dữ liệu kiểu variant sang kiểu subtype Long
CSng	Chuyển dữ liệu kiểu variant sang kiểu subtype Single
CStr	Chuyển dữ liệu kiểu variant sang kiểu subtype String

Các hàm định dạng dữ liệu (Format Functions)

Tên hàm	Mô tả
FormatCurrency	Trả về biểu thức được định dạng kiểu như currency
FormatDateTime	Trả về biểu thức được định dạng kiểu date or time
FormatNumber	Trả về biểu thức được định dạng kiểu số.
FormatPercent	Trả về biểu thức được định dạng kiểu percentage

Các hàm toán học (Math Functions)

Tên hàm	Mô tả
Abs	Giá trị tuyệt đối của một số
Atn	Trả về cotan của một số
Cos	Giá trị cosine của một số (Góc)
Hex	Cho giá trị hexadecimal của một số
Int	Trả về phần nguyên của một số
Fix	Trả về phần nguyên của một số
Log	Logarit tự nhiên của một số
Oct	Cho giá trị octal của một số
Rnd	Cho một số ngẫu nhiên nhỏ hơn 1 và lớn hơn hoặc bằng 0
Sgn	Trả về một số đại diện cho dấu của số
Sin	Giá trị Sin của một số (Góc)
Sqr	Bình phương của một số
Tan	Giá trị Tang của một số (Góc)

Các hàm về array (Array Functions)

Tên hàm	Mô tả
Array	Trả về một variant chứa một array
IsArray	Trả về giá trị Boolean cho biết biến đó có phải là một array hay không.
Join	Trả về một xâu chứa số các xâu con trong dãy
LBound	Trả về cận dưới của chiều được chỉ định của một array
Split	Trả về một array 1 chiều chứa một số lượng phần tử được chỉ định.
UBound	Trả về cận trên của chiều được chỉ định của array

Các hàm về xâu (String Functions)

Tên hàm	Mô tả
InStr	Trả về vị trí đầu tiên mà một xâu xuất hiện trong một xâu khác. Tìm kiếm được bắt đầu từ ký tự đầu tiên của xâu
InStrRev	Trả về vị trí đầu tiên mà một xâu xuất hiện trong một xâu khác. Tìm kiếm được bắt đầu từ ký tự cuối cùng của xâu
LCase	Chuyển tất cả các ký tự của một xâu thành chữ thường
Left	Trả về một xâu có độ dài được chỉ định tính từ ký tự đầu tiên
Len	Trả về đọ dài của xâu

LTrim	Xoá các ký tự trắng bên trái của xâu
RTrim	Xoá các ký tự trắng bên phải của xâu
Trim	Xoá các ký tự trắng ở cả hai phía của xâu
Mid	Trả về một xâu có độ dài được chỉ định và bắt đầu từ một vị trí được chỉ định của xâu nguồn
Replace	Thay một phần của xâu bởi một xâu khác. Số các lần thay được chỉ định trước.
Right	Trả về một xâu có độ dài được chỉ định tính từ ký tự cuối cùng
Space	Trả về một xâu chỉ gồm toàn dấu cách. Số lượng dấu cách được chỉ định
StrComp	So sánh hai xâu và trả về một giá trị là kết quả của phép so sánh
String	Trả về một xâu có đọ dài được chỉ định và được tạo ra bằng cách lặp đi lặp lại một ký tự nào đó
StrReverse	Trả về một xâu bằng cách quay ngược một xâu có sẵn
UCase	Chuyển tất cả các ký tự của 1 xâu thành chữ hoa

Các hàm khác (Other Functions)

Tên hàm	Mô tả
CreateObject	Tạo một Object có kiểu được chỉ định
Eval	Đánh giá một biểu thức và trả về một giá trị là kết quả của sự đánh giá đó
InputBox	Hiển thị một hộp thoại cho phép người sử dụng có thể điền thông tin vào
IsEmpty	Trả về một giá trị Boolean cho biết một biến đã được gán giá trị hay chưa
IsNull	Kiểm tra xem một biến có là Null (Không chứa dữ liệu) không. Kết quả là một giá trị Boolean
IsNumeric	Trả về một giá trị Boolean cho biết biểu thức đó có thể chuyển thành dạng số không
MsgBox	Hiển thị một hộp tin nhắn và chờ người sử dụng click vào một nút lệnh, và trả về giá trị cho biết người sử dụng đã click nào nút lệnh nào
Round	Làm tròn một số

ScriptEngine	Trả về tên của script đang dùng
TypeName	Trả về tên kiểu dữ liệu con của biến
VarType	Trả về giá trị của kiểu dữ liệu con của biến

Câu hỏi và Bài tập chương 3

1. Hãy tạo ra một liên kiết và lập trình script sao cho khi nhấp vào liên kết thì Windows hỏi .Nếu OK thì ta link dến trang đó ,không thì ta không là gì cả

2. Hãy tạo một chương trình máy tính điện tử như sau :

3. Tạo một chương trình mô tả Lịch để bàn như sau :

4. Viết chương trình cho phép link dến một trang Web khác trong đó cho phép tuỳ chọn các đối tượng Window

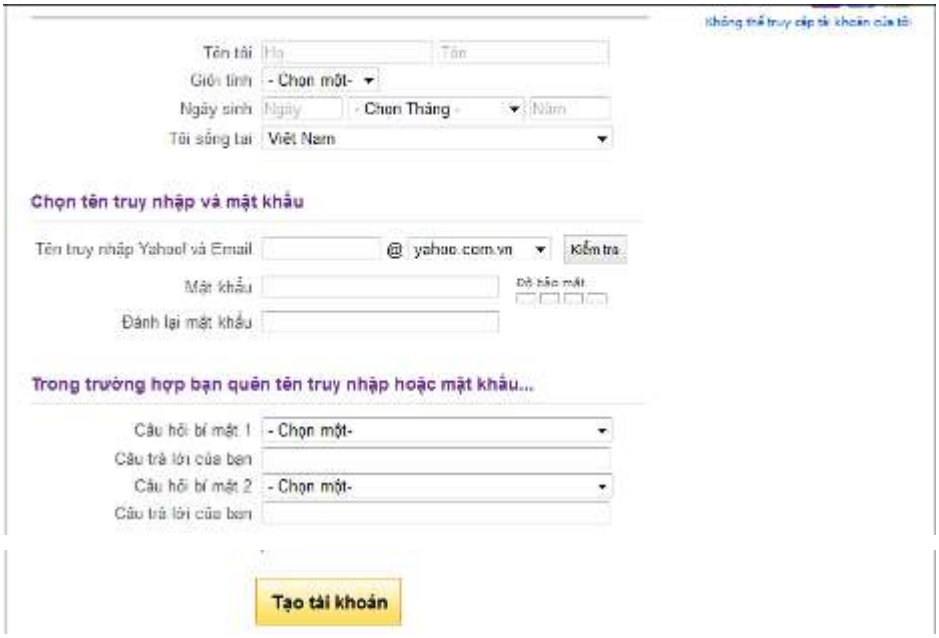
5. Tạo dòng chữ bay vào thanh trạng thái từng chữ cái một

6. Việt chương trình tạo hiệu ứng ; khi đưa chuột vào thí xuất hiện ảnh khác khi đưa ra khỏi ảnh thì hiện ảnh cũ

7. Viết chương trình Tạo nút bấm khi người dùng bấm vào thì hiện mã nguồn chương trình

8. Viết chương trình cho phép tạo ra hộp dropdown cho người dùng lựa chọn ngày tháng năm.

9. Viết chương trình cho phép kiểm tra việc nhập các thông tin trên form (như địa chỉ emai, độ dài mật khẩu, số điện thoại,…)



Chương 4: LẬP TRÌNH ASP

4.1. IIS (Internet Information Server)

4.1.1. Giới thiệu IIS

IIS là viết tắt của từ (Internet Information Services), IIS được đính kèm với các phiên bản của Windows.

Microsoft Internet Information Services (các dịch vụ cung cấp thông tin Internet) là các dịch vụ dành cho máy chủ chạy trên nền Hệ điều hành Window nhằm cung cấp và phân tán các thông tin lên mạng, nó bao gồm nhiều dịch vụ khác nhau như Web Server, FTP Server,...

Nó có thể được sử dụng để xuất bản nội dung của các trang Web lên Internet/Intranet bằng việc sử dụng "Phương thức chuyển giao siêu văn bản" - Hypertext Transport Protocol (HTTP).

Như vậy, sau khi chúng ta thiết kế xong các trang Web của mình, nếu chúng ta muốn đưa chúng lên mạng để mọi người có thể truy cập và xem chúng thì chúng ta phải nhờ đến một Web Server, ở đây là IIS.

Nếu không thì trang Web của chúng ta chỉ có thể được xem trên chính máy của chúng ta hoặc thông qua việc chia sẻ tệp (file sharing) như các tệp bất kỳ trong mạng nội bộ mà thôi.

Nhiệm vụ của IIS

Nhiệm vụ của IIS là tiếp nhận yêu cầu của máy trạm và đáp ứng lại yêu cầu đó bằng cách gửi về máy trạm những thông tin mà máy trạm yêu cầu.

Chúng ta có thể sử dụng IIS để: Xuất bản một Website của chúng ta trên Internet. Tạo các giao dịch thương mại điện tử trên Internet (hiện các catalog và nhận được các đơn đặt hàng từ người tiêu dùng). Chia sẻ file dữ liệu thông qua giao thức FTP. Cho phép người ở xa có thể truy xuất database của chúng ta (gọi là Database remote access).Và rất nhiều khả năng khác, ...

Hoạt động của IIS

IIS sử dụng các giao thức mạng phổ biến là HTTP (Hyper Text Transfer Protocol) và FPT (File Transfer Protocol) và một số giao thức khác như SMTP, POP3,... để tiếp nhận yêu cầu và truyền tải thông tin trên mạng với các định dạng khác nhau.

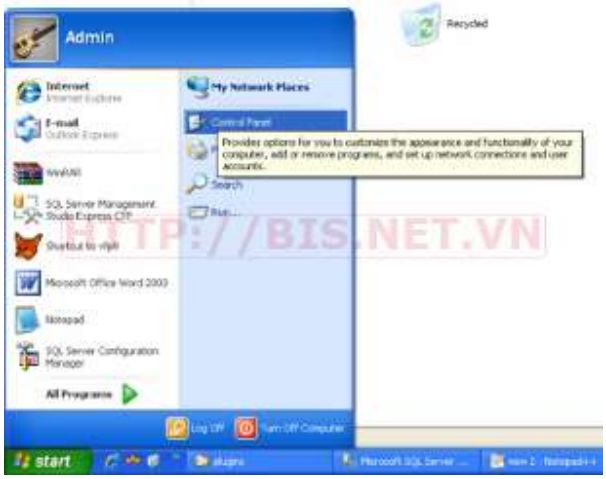
Một trong những dịch vụ phổ biến nhất của IIS mà chúng ta quan tâm trong giáo trình này là dịch vụ WWW (World Wide Web), nói tắt là dịch vụ Web.

Dịch vụ Web sử dụng giao thức HTTP để tiếp nhận yêu cầu (Requests) của trình duyệt Web (Web browser) dưới dạng một địa chỉ URL (Uniform Resource Locator) của một trang Web và IIS phản hồi lại các yêu cầu bằng cách gửi về cho Web browser nội dung của trang Web tương ứng.

Cài đặt IIS Hiện tại đã có các phiên bản 3.0, 4.0 và 5.1. Nói chung cách cài đặt không có gì khó và khác nhau lắm giữa các version. *Lưu ý* : Tốt nhất là có bản cài ngoài (từ đĩa CD hoặc download từ Internet) hoặc tham khảo bảng sau Version Cài đặt Windows 98 Cài Personal Web Server trong Add/Remove Programs trong Control Panel Window NT server 4.0 Cài Internet Information Server trong Add/Remove Programs trong Control Panel hoặc trong Windows NT Option Pack Windows 2000 Cài Internet Information Server trong Add/Remove Programs trong Control Panel Để cài IIS.

4.1.2. Cài đặt IIS.

Vào menu Start -> Control Panel



Hình 4.1. Vị trí của Control Panel

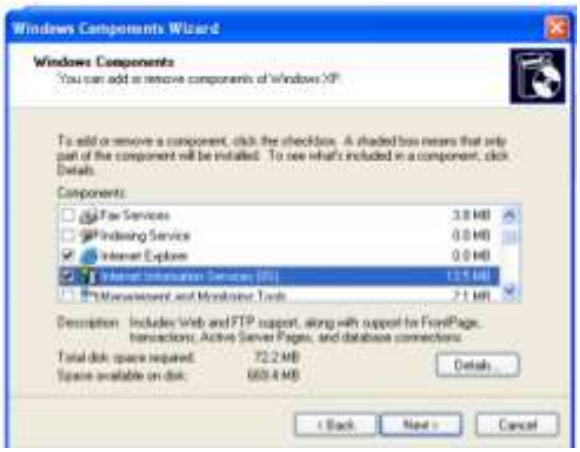
Vào mục Add, Remove Program

Hình 4.2. Vị trí của Add or Remove Program

Trong cửa sổ mở ra, tiếp tục chọn vào Add, Remove Windows Components

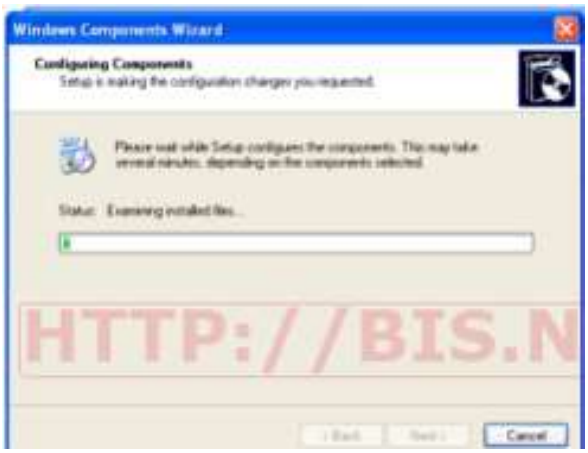
Hình 4.3. Vị trí của Windows Components

Trong cửa sổ tiếp theo, đánh dấu vào mục Internet Information Services (IIS) - đây chính là thành phần mà ta cần cài.



Hình 4.4. Vị trí của IIS

Nhấn Next.



Hình 4.5. Tiến trình cài đặt

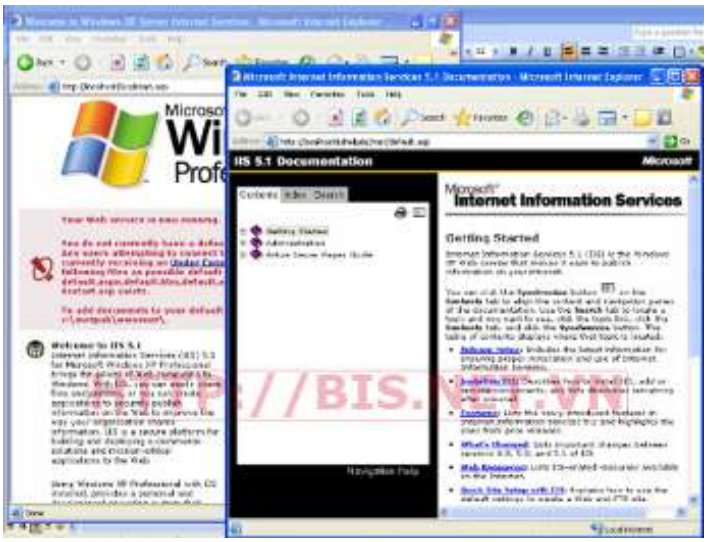
Trong quá trình cài, Windows có thể yêu cầu cho đĩa Windows vào ổ CD để nó chép các file cần thiết. Lúc này ta có thể có hai cách:

- Cho đĩa vào ổ CD và OK.
- Nếu trên máy chúng ta đã có sẵn thư mục chứa bộ cài đặt Windows, nhấn Browse và tìm đến thư mục I386 có trong bộ cài đặt đó. -> OK

Chú ý cả bộ cài cũng như đĩa Windows đều phải là bản thích hợp với phiên bản Windows hiện tại ta đang dùng. Ví dụ nếu chúng ta dùng Windows XP Service Pack 2 thì ta cũng phải bỏ đúng đĩa Windows XP Service pack 2 vào thì mới cài được. Đợi một chút để máy cài IIS vào.

Khi máy báo cài xong, nhấn Finish, đến đây ta đã cài xong IIS.

Bây giờ để biết ta cài thành công hay chưa, ta mở Internet Explorer lên. Trên dòng địa chỉ (Address), gõ vào chữ "localhost" -> nhấn Enter, nếu xuất hiện trang như dưới đây có nghĩa là ta cài thành công.



Hình 4.6. Màn hình test IIS

Như vậy chúng ta đã cài thành công IIS trên máy local. Từ đây chúng ta có thể thực hành thiết kế website trên máy của mình.

4.1.3. Thực hiện file ASP trên IIS

Sau khi start IIS mặc định web server sẽ phục vụ ở địa chỉ http://localhost (địa chỉ trên máy local, cũng giống như một địa chỉ website kiểu như http://www.yahoo.com trên Internet) Chúng ta tạo một thư mục ảo (Virtual Directory) trên web server để chứa ứng dụng web, ví dụ http://localhost/test ở đây "test" còn được gọi là Alias của Virtual Directory này. Vậy để lưu trữ các trang ASP trên server trước hết ta sẽ tạo một Virtual Directory với một Alias và thư mục tương ứng rồi upload các file ASP vào thư mục này, sau đó truy cập các trang ASP này thông qua địa chỉ http://localhost/Alias

Cách tạo một Virtual Directory trong IIS:

Vào Web Server từ Control Panel=> Administrative Tools=>Internet Services Manager (hoặc Computer Management)=> Default Website (nếu thấy nó đang stop thì start nó lên) => New=> Virtual Directory (làm theo wizard, chọn các tham số Alias: tên Virtual Directory của mình ví dụ "test", Directory: thư mục chứa Website ví dụ "C:\Web"). Sau khi kết thúc wizard này chúng ta đã có một Virtual Directory sẵn sàng trên web server. Hãy save các trang asp vào thư mục "C:\Web". Địa chỉ truycập vào website trong trường hợp này sẽ là: http://localhost/test/

Một cách khác cũng tương tự và dễ thao tác hơn là nhấn chuột phải vào thư mục C:\web, chọn Properties => Web sharing => Share this folder=> AddAlias.

4.2. ASP (Active Server Page)

4.2.1. Giới thiệu về ASP.

ASP (Active Server Page)

ASP do Microsoft xây dựng, là môi trường lập trình ứng dụng phía server, hỗ trợ cho việc xây dựng các ứng dụng web. ASP không phải phải là một công nghệ độc lập, nó cho phép liên kết nhiều ứng dụng web gồm các thành phần HTML, VBScript, SQL, ADO (ActiveX Data Objects) và COM (Component Object Model)

ASP được hỗ trợ mặc định khi cài đặt IIS(Internet Information Server), trong môi trường hệ điều hành khác cần thiết phải cài đặt thư viện hỗ trợ ASP như Sun Chili!Soft.

Những ưu điểm và nhược điểm của ASP:

Ưu điểm: Sử dụng ADO để thao tác trên cơ sở dữ liệu thuận lợi, có nhiều đối tượng dựng sẵn hỗ trợ việc lập trình dễ dàng, nhanh chóng. Đặt biệt, ASP có tính mở, nó cho phép người lập trình dùng một ngôn ngữ lập trình khác để xây dựng các component để xử dụng trong môi trường ASP.

Nhược điểm: Tốc độ thực hiện chậm hơn công nghệ Java Servlet của Sun, tính bảo mật thấp vì các mã ASP đều có thể đọc được nếu người dùng có quyền truy cập vào web server.

ASP đáp ứng các yêu cầu:

- ✓ Cho phép chúng ta soạn thảo, chỉnh sửa hay thêm bớt bất kỳ thành phần nào của trang web một cách năng động.
- ✓ Đáp ứng các truy vấn của người sử dụng hoặc dữ liệu gửi qua các form.
- ✓ Truy nhập tới bất kỳ cơ sở dữ liệu nào và trả kết quả cho trình duyệt.
- ✓ Tuỳ chỉnh trang web đáp ứng những yêu cầu khác nhau của từng người sử dụng.

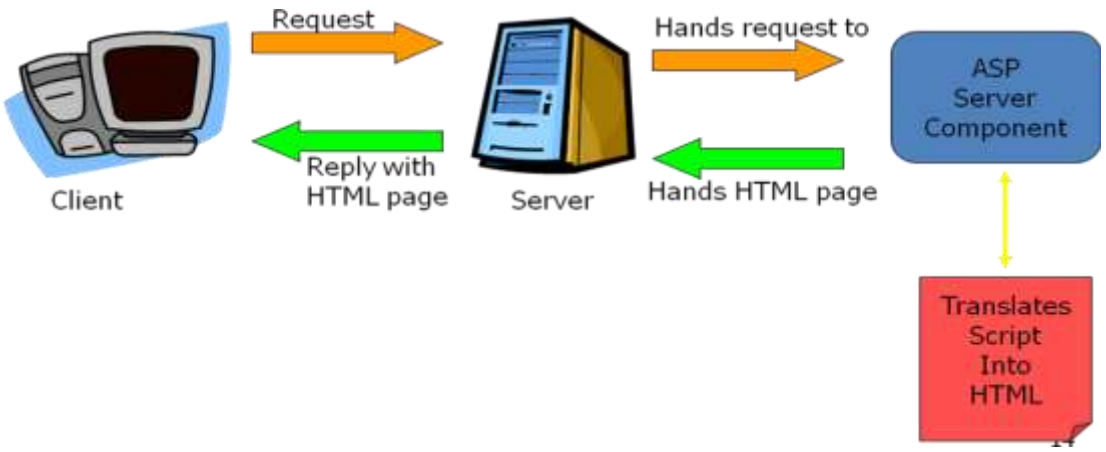
- ✓ Cấu trúc đơn giản, dễ học, dễ sử dụng, dễ phát triển, dễ sửa đổi và tốc độ xử lý nhanh. Nếu chúng ta là một người đã quen với HTML thì chúng ta có thể thấy trước đây để lấy thông tin từ một form HTML, chúng ta phải thạo một ngôn ngữ lập trình để xây dựng nên một ứng dụng theo một chuẩn gọi là CGI (Common Interface Gateway). Bây giờ, với ASP chúng ta có thể lấy được các thông tin đó một cách dễ dàng mà chỉ cần một vài dòng lệnh đơn giản nhúng trực tiếp trong trang HTML.
- ✓ Rất an toàn vì các dòng lệnh ASP không nhìn được từ trình duyệt. Vì ASP trả về file HTML thuần tuý cho nên nó có thể xem được bằng bất cứ tình duyệt nào.
- ✓ Một chương trình ASP được lập trình tốt sẽ làm giảm tải đường truyền

4.2.2. Hoạt động của ASP

Client thông qua trình duyệt của mình để yêu cầu xử lý và gửi lại kết quả một trang web (thường dưới dạng địa chỉ URL trang web) nào đó lên WEB SERVER, WEB SERVER sẽ xét xem trang đó là một trang tĩnh hay động. Nếu đó là một trang tĩnh(.html, .htm), WEB SERVER sẽ gửi lại chính file đó cho client mà không phải xử lý; nếu đó là một trang động (.ASP, .JSP, .CF...), WEB SERVER sẽ gọi một Application Server phù hợp để dịch các trang đó thông qua các bộ dịch script file (Script Engine). Trong trang ASP, chúng ta có thể sử dụng các ngôn ngữ Script tuỳ ý như: VBScript, JavaScript, Perl Script...miễn là chúng ta đã cài các Script Engine thích hợp trên WEB SERVER đó.

ASP hỗ trợ cho 2 scripts phổ dụng là: VBScript và JavaScript. Sau khi Script Engine dịch các script file, nó sẽ trả lại kết quả tới Client dưới dạng mã HTML. ASP cũng hoạt động theo mô hình trên. Sau khi Script Engine của ASP dịch trang trang ASP, nó sẽ xử lý các mã ASP và trả lại kết quả cho Client và nếu mở trang kết quả tại Client, ta sẽ không thấy còn cấu trúc lệnh của ASP trong đó.

Hình dưới đây minh hoạ quá trình hoạt động của ASP:



Hình 4.7. Mô hình hoạt động của ASP

4.2.3. Cú pháp của ASP

Trang ASP là một text file có phần mở rộng là .asp, có cấu trúc như một tệp tư liệu HTML, trong trang ASP còn các thẻ khác để nhúng các mã kịch bản để làm việc với dữ liệu như một ngôn ngữ lập trình thật sự. Khi webserver nhận được yêu cầu từ client về một trang asp, nó sẽ yêu cầu trình xử lý trang asp (ASP engine) để tạo ra một tư liệu HTML gửi về cho client. Một trang ASP thông thường gồm có các thành phần :

- Khai báo ngôn ngữ kịch bản được sử dụng <%@ language= ngôn_ngữ%>
- Các thẻ HTML Các script cho phía client được đặt trong cặp thẻ <Script> và </Script>
- Mã chương trình được đặt trong cặp thẻ <% và %>

Với cấu trúc như vậy, ta có thể xem trang asp là một tư liệu HTML được nhúng phần xử lý viết bằng mã ASP.

Ví dụ 4.1 :

```
<%@ language = VBScript%>
<HTML><HEAD><TITLE> Trang ASP </TITLE></HEAD>
<BODY>
    <P> Trang Web đầu tiên </P>
    <%
        Response.Write("Hôm nay ngày :" &Date())
    %>
</BODY>
</HTML>
```

4.2.4. Khai báo các ngôn ngữ Script trong ASP

Ngôn ngữ kịch bản mặc nhiên trong môi trường ASP là VBScript, ngoài ra có thể sử dụng các ngôn ngữ khác như Perl, Python, Jscript ... nếu webserver có cài đặt bộ xử lý ngôn ngữ này.

Để khai báo ngôn ngữ Script được sử dụng ta sử chỉ dẫn sau đặt trên đầu trang:

<%@ LANGUAGE= ngôn_ngữ script%>

Ví dụ 4.2:

```
<%@ language = JavaScript%>
<HTML><HEAD><TITLE> Trang ASP </TITLE></HEAD>
<BODY>
    <P> Sử dụng ngôn ngữ javascript </P>
    <%
        Response.Write("Hôm nay ngày :" &Date())
```

%>

</BODY>

</HTML>

Sử dụng các Server- Side VBScript và Javascript

Bởi vì Asp script được dịch và chạy trên Server, các lệnh VBScript hiển thị giao diện người sử dụng như: InputBox và MsgBox không có tác dụng. Các hàm tạo đối tượng trong VBScript như: CreateObject và GetObject sẽ được thay thế bằng hàm tạo đối tượng Server.CreateObject của ASP.

Khai báo sử dụng Server-Side Script:

<Script Language=JScript Runat=Server>

//Các lệnh Script

</Script>

hoặc

<Script Language=VBScript Runat=Server>

//Các lệnh Script

</Script>

Để tạo một chú giải:

- Trong Server-Side VBScript, sử dụng kí tự ‘
- Trong Server-Side Jscript, sử dụng kí tự // cho một dòng hoặc {} cho một đoạn

Chú ý: VBScript không phân biệt chữ hoa chữ thường, tức là một biến new var có thể được hiểu như là New_Var hay new_var. Tuy nhiên JS lại phân biệt chữ hoa chữ thường(Case Sensitive), tức là 2 biến: new_var và New_Var là 2 biến khác nhau hoàn toàn.

4.2.5. Khai báo biến trong ASP

Một biến là một vùng nhớ trong máy tính được đặt tên dùng để lưu trữ dữ liệu. Dữ liệu được lưu trong vùng nhớ đó gọi là giá trị biến.

Để khai báo một biến trong VBScript người ta dùng từ khoá Dim:

<% Dim Tên Biến %>

Ví dụ 4.3 sau đây minh hoạ cách khai báo, gán giá trị và sử dụng giá trị của một biến:

```
<%
    Dim ho_ten
    ho_ten = “Nguyen Van A”
    Response.Write(“Ho va ten cua toi la: ” & ho_ten)
 %>
```

Khai báo một dãy (array): Một array được dùng để lưu trữ những dữ liệu có liên quan tới nhau. Dưới đây là ví dụ về cách khai báo và sử dụng array:

```
<%
```

```
        Dim danh_sach(5), i
        danh_sach(0) = "Thanh Hoa"
        danh_sach(1) = "Thuy Hien"
        danh_sach(2) = "Thanh Binh"
        danh_sach(3) = "Khanh Hung"
        danh_sach(4) = "Tuan Anh"
        For i=1 to 4
        Response.Write(danh_sach(i))
        Next
    %>
```

Khai báo biến trong JScript:

<%var Tên Biến%>

4.2.6. Phạm vi hoạt động của biến

Phạm vi hoạt động của biến quyết định sự tồn tại của biến. Trong ASP có 2 loại biến là: Biến toàn cục và biến cục bộ. Một biến được khai báo bên trong một thủ tục hoặc hàm bằng từ khoá Dim, Var được gọi là biến cục bộ.

Biến cục bộ được tạo và được huỷ mỗi khi thủ tục hoặc hàm được thi hành. Nó không thể được truy nhập từ bên ngoài thủ tục hoặc hàm đó. Một biến được khai báo bên ngoài một thủ tục được gọi là biến toàn cục; Giá trị của biến này có thể truy xuất và sửa đổi bằng các lệnh Script trong trang ASP.

Chú ý: Có thể khai báo biến cục bộ và biến toàn cục trùng tên nhau.

Ví dụ sau trả lại giá trị là 1 mặc dù có 2 biến được đặt tên là Y:

```
<%
    Dim Y
    Y=1
    Call SetLocalVariable
    Response.write Y
    Sub SetLocalVariable
    Dim Y
    Y=2
    End Sub
%>
```

Trong ASP, chúng ta có thể sử dụng biến mà không cần khai báo trước. Tuy nhiên, để tránh nhầm lẫn giữa phạm vi các biến, nên khai báo các biến trước khi sử dụng.

Để khai báo các biến có thể sử dụng trong nhiều hơn 1 file asp, chúng ta cần phải khai báo biến đó là biến phiên (session) hoặc biến ứng dụng (application).

4.2.7. Các biến phiên và biến ứng dụng

Biến phiên(session) dùng để lưu thông tin về một người sử dụng, và có giá trị trong tất cả các trang web trong suốt một phiên của người sử dụng. Một phiên của người sử dụng được tính từ lúc người sử dụng bật Browser cho đến khi tắt Browser. Tuy nhiên biến phiên sẽ tự động bị xoá nếu sau một khoảng thời gian xác định người sử dụng không Refresh lại trang. Ta có thể đặt thời gian tồn tại biến phiên này trong trang ASP. Thời gian ngầm định là 20 phút.

Biến ứng dụng(Application) có giá trị đối với tất cả các trang của ứng dụng và chỉ bị xoá khi tắt ứng dụng. Biến ứng dụng là một cách tốt để lưu trữ thông tin của tất cả các người sử dụng trong một ứng dụng.

Có thể tạo ra các biến phiên và biến ứng dụng bằng 2 đối tượng: Session và Application.

Cú pháp:

<%Session("Tên biến") = Giá trị khởi tạo%>

hoặc:

<%Application("Tên biến") = Giá trị khởi tạo%>

Chúng ta sẽ đề cập chi tiết hơn về hai loại biến này trong phần đối tượng của Asp

4.2.8. Khai báo thủ tục, hàm và cách gọi

Ta có thể sử dụng 2 ngôn ngữ Script để viết trong ASP là Java Script và VB Script.

Trong VB Script:

```
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
    Sub Tên_thủ_tục(đối số)
        'Thân thủ tục
    End Sub
    Function Tên_hàm(đối số)
        'Thân hàm
        'Tên hàm = Giá trị trả lại
    End Function
</SCRIPT>
```

Cách gọi thủ tục, hàm VBScript:

```
    Call Tên thủ tục(giá trị truyền cho thủ tục)
```

Hoặc

```
    Biến = Tên hàm(giá trị truyền cho hàm)
```

Trong Java Script:

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

```
function Tên_thủ_tục(đối số)
{
  //Thân thủ tục
}
function Tên_hàm(đối số)
{
  //Thân hàm
return Giá_trị_trả_lại
}
</Script>
```

Cách gọi thủ tục, hàm Jscript:

Tên thủ tục(giá trị truyền cho thủ tục)

Hoặc

Biến = Tên hàm(giá trị truyền cho hàm)

Ví dụ 4.4: sau minh hoạ cách khai báo một thủ tục/hàm và cách gọi chúng

```
<html>
<head>
<%
  sub vbproc(num1,num2)
    Response.Write(num1*num2)
  end sub
%>
<script language="javascript" runat="server">
  function jsproc(num1,num2)
  {
    Response.Write(num1*num2)
  }
</script>
</head>
<body>
<p>Result: <%call vbproc(3,4)%></p>
<p>Result: <%vbproc 3,4%> </p>
<p>Result: <%call jsproc(3,4)%></p>
</body>
</html>
```

4.2.9. Liên kết nhiều tệp trong một tệp

Để tránh nội dung của một tệp quá dài, khó quản lý, hoặc có những nội dung được lặp đi lặp lại trong nhiều trang, ta có thể tạo những hàm, headers, footers, hay các phần tử dùng chung trong nhiều trang đó và đưa vào các tệp nhỏ rồi liên kết chúng trong tệp chính.

Chỉ dẫn #include cho phép liên kết tệp dưới dạng vật lý hay logic.

Cách sử dụng chỉ dẫn #include:

Dưới đây là một file có tên mypage.asp:

```
<html>
<body>
<h3>Words of Wisdom:</h3>
<p><!--#include file="wisdom.inc"--></p>
<h3>The time is:</h3>
<p><!--#include file="time.inc"--></p>
</body>
</html>
```

Còn đây là file wisdom.inc:

"One should never increase, beyond what is necessary,the number of entities required to explain anything."

Và đây là nội dung file time.inc

```
<%
Response.Write(Time)
%>
```

Nếu chúng ta xem mã nguồn trên trình duyệt, chúng ta sẽ thấy chúng có dạng như sau:

```
<html>
<body>
<h3>Words of Wisdom:</h3>
<p>"One should never increase, beyond what is necessary,
the number of entities required to explain anything."</p>
<h3>The time is:</h3>
<p>11:33:42 AM</p>
</body>
</html>
```

Cú pháp cho chỉ dẫn #include như sau:

```
<!--#include virtual="somefilename"-->
```

or

<!--#include file ="somefilename"-->

Từ khoá Virtual:

Sử dụng từ khoá virtual để chỉ đường dẫn bắt đầu bởi một thư mục ảo.

Nếu có một file tên header.inc nằm trong thư mục ảo có tên /html, thì ta sử dụng dòng lệnh sau để chèn nội dung file header.inc:

<!-- #include virtual=”/html/header.inc” -->

Từ khoá File

Sử dụng từ khoá file để chỉ đường dẫn tương đối. Một đường dẫn tương đối bắt đầu bởi thư mục chứa file được chèn thêm vào.

Nếu chúng ta có một file nằm trong thư mục html, và file chèn thêm vào là header.inc nằm trong html\headers thì ta có thể dùng dòng lệnh sau để chèn file header.inc:

<!-- #include file=”headers\header.inc” -->

Chú ý rằng file chèn thêm vào (headers\header.inc) có quan hệ tương đối đối với file cần chèn. Nếu như file chứa dòng lệnh #include không nằm trong thư mục html thì dòng lệnh này không làm việc. Chúng ta cũng có thể sử dụng cú pháp (..\) để chèn một file ở thư mục cấp trên.

Chú ý: Trong các phần trên chúng ta dùng file có phần mở rộng là .inc. Nếu người sử dụng muốn mở chính file .inc đó thì toàn bộ nội dung của file này sẽ được hiển thị. Nếu chúng ta không muốn nội dung của file được nhìn thấy thì chúng ta nên để file dạng .asp. Mã nguồn của file ASP không được hiển thị khi chưa dịch. Một file có thể được include trong nhiều file khác nhau, và một file có thể include trong nó một hay nhiều lần 1 file khác. Những file được include được xử lý và chèn vào trước khi các scripts được thực hiện. Script sau sẽ không làm việc vì ASP thực hiện chỉ dẫn #include trước khi nó gán giá trị cho biến:

```
<%
  fname="header.inc"
%>
<!--#include file="<%=fname%>"-->
```

Chúng ta không thể đặt #include trong lòng script, nó sẽ không làm việc.

Đoạn mã sau sẽ không làm việc:

```
<%
    For i = 1 To n
        <!--#include file="count.inc"-->
    Next
%>
```

Nhưng đoạn dưới đây thì có làm việc:

```
<% For i = 1 to n %>
        <!--#include file="count.inc" -->
<% Next %>
```

4.3. Các đối tượng cơ bản của ASP

Đối tượng là khái niệm trừu tượng nói về một ”vật thể” (hay một structure) có khả năng lưu trữ dữ liệu và thao tác trên các dữ liệu để phục vụ cho một công việc nào đó. Trong đối tượng người ta gọi các dữ liệu là các thuộc tính còn các thao tác là các phương thức. Các đối tượng trong ASP cho phép người lập trình giao tiếp, tương tác với cả server lẫn client. Trong ASP có hai loại đối tượng đó là:

Các đối tượng cơ bản: Application, Session, Server, Request, Response, ObjectContext.

Các thành phần (component) hay đối tượng xây dựng sẵn: Dictionary, FileSystemObject, AdRotator, Browser Capabilities,…

4.3.1. Đối tượng Request

Đối tượng Request nhận tất cả các thông tin mà trình duyệt của client gởi đến web server thông qua một yêu cầu HTTP (HTTP request), các thông tin này được lưu trữ trong 5 kiểu tập hợp của đối tượng Request. Các biến có thể được truy cập trực tiếp bằng cách gọi Request(Var), không thông qua tên tập hợp; trong trường hợp này Web server tìm thông tin trong các tập hợp theo thứ tự: QueryString, Form, Cookies, Client, Certificate và ServerVariable. 5 collection cho phép chúng ta truy xuất tất cả các loại thông tin về yêu cầu của browser đối với server.

1) Các collection của đối tượng Request:

- **Client Cirtificate**

Một tập các giá trị của tất cả các trường (field) hay các mục (entry) trong Client certificate mà browser chuyển đi để trình cho server khi truy xuất một trang hay tài nguyên. Các thành phần của tập đều là giá trị chỉ đọc (read- only).

- **Cookies**

Cookies là một file văn bản có kích thước nhỏ được lưu trữ trên máy client. Mỗi khi người dùng thăm một Website, ta có thể bí mật gắn một tập tin chứa các thông tin mà mình muốn lên đĩa cứng của họ, chẳng hạn như thông tin về user, thông tin về số lần truy cập website,... Tuy nhiên các Cookies không phải được truy cập ngẫu nhiên bởi các Website mà chúng được truy cập bởi các domain tạo ra chúng.

Các cookies trong đối tượng Request đều là thuộc tính chỉ đọc (read-only) do đó ta chỉ có thể xem các giá trị cookies mà không thể sửa đổi giá trị của chúng. Để lấy giá trị của cookies ta sử dụng

Cú pháp:

Request.Cookies(name)[(key)|.attribute]

Trong đó:

- ✓ name: tên của cookie (kiểu chuỗi)
- ✓ key: khóa của cookie cần lấy giá trị (kiểu chuỗi)
- ✓ attribute: thông tin của cookie, là một trong các thông số sau:

- Domain: (chỉ đọc - read only) cookie chỉ được gởi cho đối tượng Request của domain này.
- Expires: (chỉ ghi - write only) chỉ định ngày mà Cookies hết hiệu lực (expires), nếu không chỉ định ngày thì cookie sẽ expires khi kết thúc phiên làm việc.
- HasKeys: (chỉ đọc - read only) xác định khóa của cookie có tồn tại không.
- Path: (chỉ ghi- write only) nếu thuộc tính này được xác lập thì chỉ cookie chỉ được gởi cho những Request của đường dẫn này, nếu không thì cookie chỉ được gởi cho những Request thuộc đường dẫn của ứng dụng.
- Secure (chỉ ghi-write only) xác định cookie có bảo mật hay không.

Một cookie có thể chứa đựng một tập hợp các giá trị. Ta nói cookie đó có nhiều khóa.

- ➢ **Form**

Các Form cho phép người dùng nhập vào dữ liệu thông qua các control HTML như edit, radio button, check box, ... Khi người dùng submit một biểu mẫu thì tất cả các giá trị của các control trong phân đoạn <FORM> sẽ được gởi lên Web Server khi đặt giá trị của thuộc tính METHOD trong tag <FORM > là POST. Các thành phần của đối tượng này đều là giá trị chỉ đọc (read only).

Để truy xuất các giá trị của các control HTML mà người dùng submit bằng phương thức POST ta sử dụng

Cú pháp sau:

Request.Form(controlname)

Trong đó controlname là tên của control mà ta cần lấy giá trị.

Ví dụ 4.5:

```
<HTML>
<BODY>
Chao ban:
    <%
        Response.Write (Request.Form("Ho_Lot"))
        Response.Write (" " & Request.Form("Ten"))
    %>
</BODY>
</HTML>
```

Trong ví dụ này ta đọc ra 2 giá trị từ 2 control trên form gửi về qua phương thức Post

- **QueryString**

Khi người dùng yêu cầu một trang hay đệ trình (submit) một biểu mẫu với phương thức GET thì tất cả các control HTML trong phân đoạn <Form> của biểu mẫu sẽ được Browser gắn vào URL theo từng cặp tên/giá trị.

QueryString được dùng để lấy về các giá trị trong một biểu mẫu với phương thức là GET. Tất cả các thông tin được gởi từ biểu mẫu với phương thức GET sẽ được gắn vào URL trên thanh address của browser và do đó mọi người có thể thấy được các thông tin này, tuy nhiên lượng thông tin được gởi này có giới hạn. Các thành phần của tập đều là giá trị chỉ đọc (read-only).

Để truy xuất các giá trị của các control HTML mà người dùng submit bằng phương thức GET ta sử dụng

Cú pháp sau:

Request.QueryString (controlname)

Tương tự như trên controlname là tên của đối tượng điều khiển trên form cần lấy giá trị.

Ví dụ 4.6: giống như ví dụ trên trong trường hợp người dùng gửi thông tin trên form bằng phương thức get thì ta đọc như sau

```
<HTML>
<BODY>
Chào bạn:
<%
    Response.Write (Request.QueryString("Ho_Lot"))
    Response.Write (" " & Request.QueryString("Ten"))
%>
</BODY>
</HTML>
```

- **ServerVariables**

Khi cần lấy giá trị các biến môi trường của Server ta dùng tập ServerVariables.

Cú pháp:

Request.ServerVariables (variable)

Với variable chỉ định giá trị gì ta cần lấy. Sau đây là một số giá trị tiêu biểu của variable

Biến	Mô tả
ALL_HTTP	Trả về tất cả các header mà client đã gửi, luôn luôn theo sau HTTP_ và viết hoa

AL_RAW	Trả về tất cả các header ở dạng thô
APPL_MD_PATH	Trả về đường dẫn cho ứng dụng dùng cho DLL ISAPI
APPL_PHYSICAL_PATH	Trả về đường dẫn vật lý tương ứng của đường dẫn
AUTH_PASSWORD	Trả về giá trị đã nhập vào trên hộp thoại xác nhận của client
AUTH_TYPE	Cách thức mà server dùng để kiểm tra xác nhận người dùng
AUTH_USER	Trả về tên của người dùng (username)
CERT_COOKIE	Trả về ID duy nhất của client
CONTENT_LENGTH	Trả về kích thước của dữ liệu mà client gửi
CONTENT_TYPE	Trả về kiểu dữ liệu
HTTP_ <*headername*>	Trả về giá trị chứa trong header *Headername*
HTTP_USER_AGENT	Trả về một chuỗi mô tả browser gởi yêu cầu
LOCAL_ADDR	Trả về địa chỉ của server mà browser gởi yêu cầu tới

Ví dụ 4.7: Chúng ta có thể dùng vòng lặp để xem tất cả các biến của server như sau:

```
<%
    For each x in Request.ServerVariables
       Response.Write (x & "<BR>")
    Next
%>
```

2) Thuộc tính (Property) của đối tượng Request

Đối tượng Request chỉ có một thuộc tính duy nhất đó là TotalBytes. Thuộc tính TotalBytes là thuộc tính chỉ đọc (read-only), nó trả về số byte dữ liệu mà người dùng chuyển lên server.

Ví dụ 4.8:

```
<%
    Dim ByteCount
    ByteCount = Request.TotalBytes
    Response.Write("ByteCount = " & ByteCount & " bytes")
%>
```

Đoạn chương trình này sẽ in ra màn hình tổng các byte dữ liệu mà một form người dùng gửi lên Server. Kết quả in ra trang web phụ thuộc vào dữ liệu cụ thể trong Form.

Ví dụ nhập: Name = Quan, Age = 56, và Sex = male.

Ta có : ByteCount = 39 bytes

3) Phương thức (Method) của đối tượng Request

BinaryRead(Count) : trả về số byte đã được gởi đến web server từ browser như là một phần của POST request. Khi phương thức này được thực hiện xong, biến Count trả về số byte đã được đọc. Giá trị của count tương đương với Request.TotalBytes.

Ví dụ 4.9 :

```
<%
    Dim ByteCount, BinRead
    ByteCount = Request.TotalBytes
    BinRead = Request.BinaryRead(ByteCount)
    ...
    Response.BinaryWrite(BinRead)
%>
```

Ví dụ 4.10 sau đây sử dụng các tập hợp của đối tượng request để lấy thông tin từ 1 form người dùng gửi về.

Tạo trang login.htm như sau

```
<html>
<head>
<meta http-equiv="Content-Language" content="en-us">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Form login</title>
</head>
<body>
<form method="POST" action="login.asp">
  <table border="0" width="43%" id="table1" bgcolor="#C0C0C0">
    <tr>
      <td colspan="2">
      <p align="center">Form login</td>
    </tr>
    <tr>
      <td      width="21%">Username</td>
      <td      width="48%"><input type="text" name="T1"
size="29"></td>
    </tr>
    <tr>
```

```
            <td       width="21%">Password</td>
            <td       width="48%"><input type="password" name="T2"
    size="29">
            </td>
        </tr>
        <tr>
            <td       colspan="2">
            <p   align="center"><input type="submit" value="Login"
    name="B1"><input type="reset" value="Cancel"
name="B2">
            </td>
        </tr>
    </table>
  </form>
  </body>
  </html>
```

Tạo trang login.asp để đọc thông tin của người dùng gửi về theo phương thức post như sau :

```
  <html>
  <head>
  <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
  <title>New Page 1</title>
  </head>
  <body>
  <%
     'lay thong tin tu nguoi dung ve
     dim username
     dim password
     username = Request.Form("T1")
     password = Request.Form("T2")
     response.write("Chào bạn : " & username & " Mật khẩu của bạn là: " &
  password)
   %>
  </body>
  </html>
```

Ghi và chạy chương trình ta được kết quả sau :

Hình 4.8. Kết quả chạy file login.htm

Khi ta kích login chương trình sẽ gọi đến login.asp kết quả như sau

Hình 4.9. Kết quả chạy file login.asp

4.3.2. Đối tượng Response

Đối tượng Response dùng để gửi tất cả các thông tin vừa được server xử lý trả về cho client. Chúng ta có thể xử dụng đối tượng này để giao tiếp với người sử dụng.

Đối tượng Response có nhiều thuộc tính như : Status, Expires, ContentType, CodePage, Charset,… các phương thức như : Write, Redirect, End, Clear, Flush, AddHeader, BinaryWrite và có một collection cookies. Sau đây chúng ta sẽ tìm hiểu chi tiết một số thuộc tính phương thức này.

1) Các tập hợp (Collection) của đối tượng Response

Tập hợp của đối tượng Response chỉ có cookies. Đối tượng Response có thể xác lập giá trị của bất kỳ cookies nào mà ta muốn đặt trên hệ thống của client. Nếu cookies không tồn tại trên client thì nó sẽ được tạo ra.

Cú pháp :

Response.Cookies(name) [key | .Attribute] = value

Trong đó :

- name : tên cookies muốn tạo ở máy client

- key : là tham số tuỳ chọn, nếu giá trị này được thiết lập thì cookies được xem như một cookies từ điển

- Attribute : bao gồm những thông tin liên quan đến cookie như : Domain, đây là thuộc tính chỉ ghi cho biết domain tạo cookie này. Expires : quy định ngày hết hạn cookie. Haskeys: cho biết cookie có chứa item hay không.

Ví dụ 4.11:

```
<%
For Each objCK In Request.Cookies
   Response.Write objCK & " = " & Request.Cookies(objCK) & "<br>"
Next
   %>
```

2) Thuộc tính (Property) của đối tượng Response

Đối tượng Response cung cấp một số thuộc tính sau đây :

- Buffer=True|False
- CacheControl "setting"
- Charset = "value"
- ContentType="kiểu-MINE"
- Expires minutes
- ExpiresAbsolute #date[time]#
- IsClientConnected
- Status = "code message"

Buffer: Dùng để xác định xem kết quả được tạo ra bởi trang ASP có được giữ lại trong vùng đệm hay không. Thuộc tính Buffer nhận một trong 2 giá trị là true hoặc false. Nếu nhận giá True thì kết quả được tạo ra bởi trang ASP sẽ được server giữ trong vùng đệm cho đến khi tất cả các script của trang được xử lý xong, hay đến khi phương thức **Flush** hoặc phương thúc End được gọi. Giá trị này cần được xác lập trước tag trong tập tin .asp. Còn nếu thuộc tính Buffer nhận giá trị False thì kết quả sẽ được gởi đi ngay khi nó được xử lý.

Cú pháp:

Response.Buffer [= true | false]

CacheControl: Thuộc tính này dùng để xác định xem proxy server có thể cất giữ kết quả được tạo ra bởi ASP hay không. Mặc định thì proxy sẽ không cất giữ. CacheControl chỉ có thể nhận một trong hai giá trị đó là "public hoặc "private". Nếu đặt thuộc tính này là "private" thì chỉ những vùng cache riêng mới có thể giữ còn proxy server sẽ không lưu trữ những trang này. Còn nếu đặt thuộc tính này là "public" thì proxy sẽ cất giữ những trang này.

Ví dụ 4.12: Chúng ta có thể thiết lập như sau:

<% Response.CacheControl = "Public"%>

hoặc

<% Response.CacheControl = ”Private”%>

Charset: Đây là thuộc tính kiểu chuỗi, thuộc tính này ghép tên của tập ký tự vào vùng context-type của đối tượng Response. Thuộc tính này chấp nhận bất cứ chuỗi ký tự nào bất chấp chuỗi đó đúng hay sai. Giá trị mặc định là ISO-LATIN-1

Cú pháp:

Response.Charset (charsetname)

Ví dụ 4.13:

<% Response.Charset = ”ISO-8859-1”%>

ContentType: Đây là thuộc tính kiểu chuỗi, thuộc tính này đặt kiểu hiển thị của nội dung HTTP cho đối tượng Response.

Nếu một trang ASP không chỉ định thuộc tính ContentType thì content-type mặc định sẽ là: content-type:text/html

Cú pháp:

Response.ContentType [= contenttype]

Sau đây là một vài giá trị contenttype thông dụng:

<% Response.ContentType = ”text/HTML”%>

<% Response.ContentType = ”image/GIF”%>

<% Response.ContentType = ”image/JPEG”%>

<% Response.ContentType = ”text/plain”%>

Ví dụ 4.14: Đoạn chương trình sau đây sẽ mở một spreadsheet trên browser (nếu chúng ta đã cài đặt Excel vào máy)

<%Response.ContentType = ”application/vnd.ms-excel”%>

Expires: Thuộc tính Expires đặt thời gian bao lâu (tính theo phút) một trang sẽ được cất giữ ở browser trước khi nó hết hạn (expire). Nếu người dùng quay lại trang đó trước khi nó hết hạn thì trang đã cất giữ trước đó sẽ được hiển thị lên. Nếu ta muốn một trang không bao giờ hết hạn thì ta đặt thuộc tính Expire là -1.

Cú pháp:

Response.Expires [= number]

Ví dụ 4.15: Nếu ta muốn cho một trang sẽ hết hạn sau 24 giờ (= 1400 phút) ta đặt như sau:

<%

Respose.Expires = 1440

%>

ExpiresAbsolute: Tương tự như thuộc tính Expires, thuộc tính ExpiresAbsolute đặt một ngày và giờ xác định mà một trang được cất giữ trên browser sẽ hết hạn. Nếu ta chỉ định thời gian mà không chỉ định ngày cụ thể thì trang sẽ hết hạn tại giờ chỉ định vào ngày

mà script được thực thi. Còn nếu ta chỉ định ngày mà không chỉ định thời gian thì trang được browser cất giữ sẽ bị hết hạn vào lúc nửa đêm của ngày chỉ định.

Cú pháp:

Response.ExpiresAbsolute [= [date][time]]

Ví dụ 4.16: Đoạn mã sau đây chỉ định rằng trang sẽ hết hạn vào lúc 4h00 chiều ngày 11 tháng 10 năm 2003:

```
<%
  Respose.ExpiresAbsolute = #October 11,2003 16:00:00#
%>
```

IsClientConnected: Thuộc tính này xác định xem client có còn nối kết (connect) với server hay không. Thuộc tính này mang 1 trong 2 giá trị đó là true hoặc false. Mang giá trị true nếu client còn kết nối tới server và mang giá trị false trong trường hợp ngược lại.

Cú pháp:

Response.IsClientConnected

Ví dụ 4.17: Đoạn code sau đây kiểm tra người dùng còn kết nối hay không?

```
<%
    If Response.IsClientConnected = true then
            Response.Write (”Nguoi dung con connect!”)
    Else
            sponse.Write (”Nguoi dung khong con connect!”)
    End If
%>
```

Pics: Thuộc tính này thêm một giá trị vào nhãn PICS ở phần header của đối tượng Response.

Ví dụ 4.18:

```
<%
Response.PICS ("(PICS-1.1
by " & chr(34) & "xyz@yahoo.com" & chr(34) &
" for " & chr(34) & "http://www.XXX.com" & chr(34) &
" on " & chr(34) & "2002.10.05T02:15-0800" & chr(34) &
" r (n 2 s 0 v 1 l 2))")
%>
```

Status: Thuộc tính này chỉ định giá trị của dòng trạng thái mà server trả về cho client và ta có thể dùng thuộc tính này để chỉnh sửa dòng trạng thái đó. Giá trị của dòng trạng thái bao gồm: ba con số dầu tiên là mã trạng thái và mô tả chi tiết của mã trạng thái đó (chẳng hạn như: 404 Not Found).

Cú pháp:

Response.Status = statusdescription

với statusdescription là dòng mô tả trạng thái

Ví dụ 4.19: Đoạn code sau đây sẽ kiểm tra quyền của user dựa vào địa chỉ của họ

```
<%
    Dim IP
    IP = Request.ServerVariables("REMOTE_ADDR")
    If IP <> 172.16.20.99"" Then
      Response.Status = "401 Unauthorized"
      Response.Write (Response.Status)
      Response.End
    End If
%>
```

3) Phương thức (Method) của đối tượng Response

AddHeader: Phương thức AddHeader thêm một header HTTP mới và một giá trị cho HTTP response. Một khi một header được thêm vào thì ta không thể gỡ bỏ nó ra.

Trong IIS 4.0, chúng ta phải gọi phương thức này trước bất kỳ kết quả nào gửi tới browser. Trong IIS 5.0 chúng ta có thể gọi phương thức AddHeader tại bất cứ nơi nào trong script nhưng phải đứng trước bất cứ lời gọi hàm Response.Flush nào trong trang.

Cú pháp:

Response.AddHeader name, value

Trong đó name là tên của header còn value là giá trị của header

Ví dụ 4.20:

```
<%
  Response.AddHeader "Cảnh báo", "Máy của bạn có Virus"
%>
```

Chú ý: Tên của header không được chứa dấu gạch dưới.

AppendToLog : Phương thức này thêm một chuỗi vào cuối mục log của trình chủ. Chúng ta có thể gọi phương thức này nhiều lần trong một script, mỗi lần gọi sẽ gắn thêm một chuỗi vào mục log của trình chủ.

Cú pháp:

Response.AppendToLog (string)

Ví dụ 4.21:

```
<%
  Response.AppendToLog "Client co virus!"
%>
```

Chú ý: Chuỗi cần ghi vào mục log không được chứa bất kỳ dấu phẩy (,) nào.

BinaryWrite: Phương thức này ghi dữ liệu trực tiếp xuống browser mà không phải chuyển đổi bất kỳ ký tự nào. Phương thức này thường được dùng để ghi dữ liệu ảnh (BLOB) từ cơ sở dữ liệu xuống browser.

Cú pháp:

Response.BinaryWrite (data)

Clear : Phương thức này xóa tất cả các kết xuất HTML được trình chủ đưa vào vùng đệm. Nhưng phương thức này không xóa phần header của đối tượng Response mà chỉ xóa phần nội dung của đối tượng Response. Nếu thuộc tính Buffer của đối tượng Response được đặt là false thì phương thức này sẽ gây ra lỗi lúc thi hành (vì không có vùng buffer thì lấy gì mà xóa!!!).

Cú pháp:

Response.Clear

Ví dụ 4.22:

```
<%
Response.Buffer = true
%>
Đây là phần nội dung của trang Web. Nội dung này sẽ được gửi tới người dùng
Bắt đầu xóa Buffer
<%
Response.Clear
%>
```

Kết quả khi duyệt trang web này là người dùng không thấy gì cả (vì trang HTML mà Server đưa vào trong vùng đệm chưa kịp gửi đã bị xóa bởi việc gọi phương thức clear.)

End : Phương thức này dùng để dừng việc xử lý một script và trả về kết quả hiện tại. Nếu thuộc tính Buffer được đặt là true thì khi gọi phương thức này Server sẽ gửi các kết xuất HTML được lưu trong vùng đệm xuống browser. Nếu ta không muốn đưa kết quả xuống cho browser thì ta gọi phương thức clear trước khi gọi phương thức này.

Cú pháp:

Response.End

Ví dụ 4.23: Đoạn văn bản này sẽ được gửi tới browser và người dùng có thể đọc được:

```
<%
  Response.End
%>
Đoạn văn bản này sẽ không được gửi và đã gọi phương thức End rồi.
```

Flush: Gọi phương thức này để chuyển các kết xuất HTML mà Server lưu giữ lại trong vùng đệm xuống browser ngay lập tức. Nếu thuộc tính Buffer được đặt là false thì thuộc tính này sẽ gây ra lỗi lúc thi hành. Ý nghĩa của phương thức là gửi tất cả dữ liệu trong vùng đệm cho client. Muốn sử dụng phương thức này phải thiết lập Response.Buffer = True.

Cú pháp:

Response.Flush

Redirect: Phương thức này dùng để chuyển người dùng đến một trang khác được chỉ định trong đường dẫn URL.

Cú pháp:

Response.Redirect (URL)

Ví dụ 4.24: sau đây minh họa việc đăng nhập của người dùng.

Tạo tập tin login.htm với nội dung sau:

```
html>
<head>
<meta http-equiv="Content-Language" content="en-us">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Form login</title>
</head>
<body>
<form method="POST" action="login.asp">
   <table border="0" width="43%" id="table1" bgcolor="#C0C0C0">
       <tr>
           <td colspan="2">
           <p align="center">Form login</td>
       </tr>
       <tr>
           <td width="21%">Username</td>
           <td width="48%"><input type="text" name="UserName"
   size="29"></td>
       </tr>
       <tr>
           <td width="21%">Password</td>
           <td width="48%"><input type="password" name="Password"
   size="29"></td>
       </tr>
```

```
        <tr>
            <td colspan="2">
            <p align="center"><input type="submit" value="Login"
    name="B1"><input type="reset" value="Cancel"
name="B2"></td>
        </tr>
    </table>
</form>
</body>
</html>
```

Tạo tập tin login.asp với nội dung sau:

```
<%
Dim User, Pass
User = Request.Form("UserName")
Pass = Request.Form("Password")
If (User = "sv") and (Pass = "1234") Then
  Response.Redirect "success.asp"
Else
  Response.Redirect "login.htm"
End If
%>
```

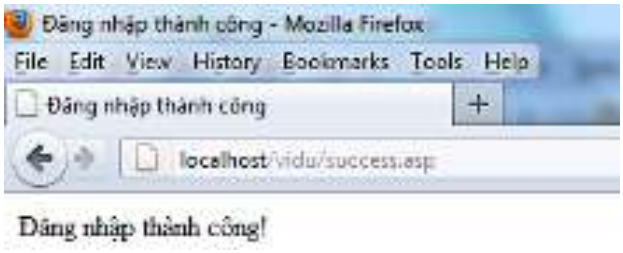
Tạo trang success.asp như sau :

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Đăng nhập thành công</title>
</head>
<body>
    Đăng nhập thành công!
</body>
</html>
```

Kết quả khi đăng nhập thành công trang success.asp sẽ được gọi như sau :

Hình 4.10. Kết quả chạy file login.htm

Kích vào login :



Hình 4.11. Kết quả chạy file success.asp

Write: Phương thức này dùng để ghi dữ liệu ra tập tin kết xuất dạng HTML để gửi cho browser. Dữ liệu này có thể là số, chuỗi, ngày, ...

Cú pháp:

Response.Write (text)

Ví dụ 4.25:

```
<%
    Response. Write ( “Chào bạn đến với ASP!” & “”)
    Dim x
    x = 100
    Response.Write x
%>
```

4.3.3. Đối tượng Cookies

1) Giới thiệu

Có lẽ chúng ta cũng đã từng đăng ký là một thành viên của một trang web hay một forum nào đó, và chắc cũng không ít lần ngạc nhiên khi chúng ta vừa yêu cầu đến một trang web hay forum mà chúng ta đã đăng ký trước đó, trang web nhận ngay ra, chúng ta chính là thành viên của họ và gửi ngay lời chào đến chúng ta, chẳng hạn: Chào Nguyễn Anh Tài.

Làm sao mà Web Server nhận ra được mình nhỉ? Mình đã đăng ký từ ngày hôm qua kia mà? Không đâu xa cả, những thông tin đó được lưu ngay chính tại máy của chúng ta. Những thông tin được Web Server lưu tại máy Client được gọi là Cookies.

Không giống như đối tượng Session, đối tượng Cookies cũng được dùng để lưu trữ thông tin của người dùng, tuy nhiên, thông tin này được lưu ngay tại máy gửi yêu cầu đến Web Server. Có thể xem một Cookie như một tập tin (với kích thước khá nhỏ) được Web Server lưu tại máy của người dùng. Mỗi lần có yêu cầu đến Web Server, những thông tin của Cookies cũng sẽ được gửi theo về Server.

2) Làm việc với Cookies

- **Thêm Cookies**

Để thêm cookies ta sử dụng phương thức sau :

Response.Cookies.Add(<HttpCookie>)

Ví dụ 4.26:

```
<%
Dim cookTen_dn As New HttpCookie("Ten_dang_nhap")
cookTen_dn.Value = txtTen_dang_nhap.Text
cookTen_dn.Expires = Date.Today.AddDays(1)
Response.Cookies.Add(cookTen_dn)
%>
```

Trong ví dụ trên, chúng ta đã tạo ra Cookies có tên là Ten_dang_nhap lưu trữ tên đăng nhập của người dùng. Thông tin này sẽ được lưu trữ trên Cookies 1 ngày kể từ ngày hiện hành trên Web Server.

- **Lấy giá trị từ Cookies**

Để lấy giá trị từ một cookies ta có thể sử dụng phương thức sau :

Dim <biến Cookie> As HttpCookie

<biến Cookie> = Request.Cookies(<Tên Cookies>)

<biến Cookie>.Value ' Giá trị của Cookies

Trong trường hợp Cookies chưa được lưu hoặc đã hết thời hạn duy trì tại Client, giá trị nhận được là Nothing.

4.3.4. Đối tượng Session

Khi chúng ta có nhu cầu truyền giá trị từ trang asp này sang trang asp khác trong một phiên làm việc, chúng ta sử dụng đối tượng Session.

Đối tượng Session là đối tượng được web server tự động khởi tạo khi trang web đầu tiên của ứng dụng được người sử dụng gọi, đối tượng Session được dùng để lưu lại các thông tin của các người dùng tại một phiên làm việc như tên, id v.v., đối tượng này sẽ tồn tại cho đến khi bị huỷ hay hết hiệu lực (time out).

Bằng cách sử dụng phương thức và thuộc tính của đối tượng này, chúng ta có thể khởi tạo, gán giá trị, truy cập và hủy đối tượng này trong một phiên làm việc nhằm quản lý

người sử dụng khi họ truy cập vào website. Sau đây chúng ta sẽ đi tìm hiểu một số thuộc tính, phương thức và tập hợp của đối tượng này giúp ta lập trình quản lý người dùng trên web.

1) Tập hợp của đối tượng Session

- **Contents**

Tập hợp Contents chứa tất cả các phần tử đã được gắn thêm vào đối tượng Session trong quá trình thực thi script.

Cú pháp:

Session.Contents (key)

Trong đó key là tên của phần tử cần lấy.

Ví dụ 4.27: sau đây liệt kê tất cả các session đã được dùng trong ứng dụng:

```
<HTML>
    <BODY>
      <CENTER> Các session trong tập Contents </CENTER>
      <%
          Dim x
          For each x in Session.Contents
             Response.Write (x & "=" & Session.Contents (x) & "<BR>")
          Next
      %>
    </BODY>
</HTML>
```

- **StaticObjects**

Tập hợp StaticObjects chứa tất cả các đối tượng gắn vào session với thẻ HTML <object>

Cú pháp:

Session.StaticObjects(key)

Ví dụ 4.28: Đoạn chương trình sau đây hiển thị tất cả các đối tượng trong tập StaticObjects

```
<HTML>
    <BODY>
      <center> Các đối tượng trong tập StaticObjects</center>
      <%
          Dim x
          For each x in Session.Contents
             Response.Write (x & "<br>")
```

```
        Next
    %>
    </BODY>
</HTML>
```

2) Các thuộc tính của đối tượng Session

- **CodePage**

Thuộc tính CodePage cho biết tập ký tự sẽ được dùng để hiển thị nội dung của trang asp khi trình bày kết quả cho người sử dụng Session.CodePage có tác dụng cho một phiên làm việc với cú pháp như sau.

Cú pháp:

Session.CodePage(= codepage)

Sau đây là một vài giá trị CodePage và mô tả của chúng

1251 - American English and most European languages

932 - Japanese Kanji

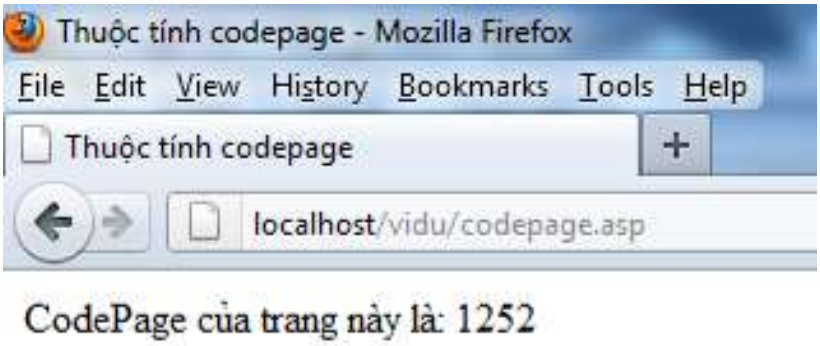
65001 – Tiếng việt Unicode

…

Ví dụ 4.29: Đoạn chương trình sau đây hiển thị codepage của một trang.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Thuộc tính codepage</title>
</head>
<body>
CodePage của trang này là:
<%
   Response.write (Session.codepage)
%>
</body>
</html>
```

Khi chạy chương trình sẽ cho ta kết quả là :



Hình 4.12. Kết quả chạy file codepage.asp

- **LCID**

Ta dùng thuộc tính LCID để thiết lập hay nhận về một con số nguyên mà nó xác định một vùng nào đó. Dữ liệu ngày, giờ và tiền tệ sẽ được hiển thị dựa theo vùng đó.

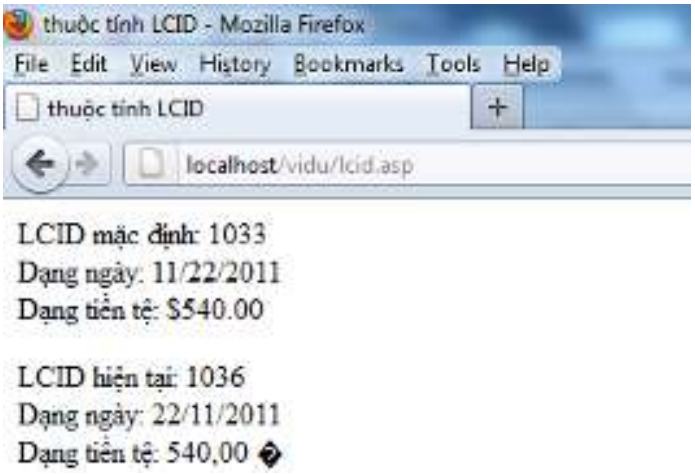
Cú pháp:

Session.LCID(= LCID)

Ví dụ 4.30: trong ví dụ này sẽ đọc ra LCID mặc định, các định dạng mặc đinh và thiết lập LCID mới với các định dạng tương ứng.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>thuộc tính LCID</title>
</head>
<body>
   <%
      Response.Write ("LCID mặc định: " & Session.LCID & "<br>")
      Response.Write ("Dạng ngày: " & date() & "<br>")
      Response.Write ("Dạng tiền tệ: " & FormatCurrency(540)& "<BR>")
      Session.LCID = 1036
      Response.Write ("<p>")
      Response.Write ("LCID hiện tại: " & Session.LCID & "<BR>")
      Response.Write ("Dạng ngày: " & date() & "<br>")
      Response.Write ("Dạng tiền tệ: " & FormatCurrency(540)& "<BR>")
      Response.Write ("</P>")
   %>
</body>
</html>
```

Kết quả chạy chương trình là:



Hình 4.13. Kết quả chạy file LCID.asp

- **SessionID**

Thuộc tính SessionID trả về một con số id duy nhất dùng để nhận diện cho mỗi người dùng. Con số này được server tạo ra và chúng ta không thể thay đổi giá trị này được.

Cú pháp:

Session.SessionID

Ví dụ 4.31: Đoạn chương trình sau đây hiển thị ra màn hình con số ID của người dùng hiện hành.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Hiển thị ID người dùng</title>
</head>
<body>
   Số ID của bạn là:
   <%
      Response.Write (Session.SessionID)
   %>
</body>
</html>
```

Kết quả chạy chương trình là :

Hình 4.14. Kết quả chạy file idnguoidung.asp

- **TimeOut**

Thuộc tính này dùng để thiết lập hay nhận về khoảng thời gian hiệu lực dành cho đối tượng Session trong ứng dụng (tính theo phút). Nếu người dùng không refresh hoặc yêu cầu một trang trong khoảng thời gian hiệu lực đó thì session sẽ kết thúc. Mặc định thời gian còn hiệu lực cho một trang là 20 phút.

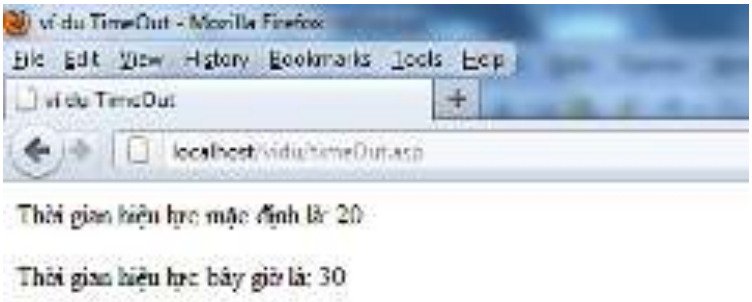
Cú pháp:

Session.TimeOut [=number]

Ví dụ 4.32: sau sẽ tạo đọc thời gian mặc định và thiết lập lại thời gian kết thúc cho session:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>ví dụ TimeOut</title>
</head>
<body>
   <P>
   Thời gian hiệu lực mặc định là:
   <% Response.Write (Session.TimeOut) %>
   </P>
   <% Session.TimeOut = 30%>
   <P>
   Thời gian hiệu lực bây giờ là:
   <% Response.Write (Session.TimeOut) %>
   </P>
</body>
</html>
```

Kết quả chạy chương trình là :



Hình 4.15. Kết quả chạy file TimeOut.asp

3) Các phương thức của đối tượng Session

- **Abandon**

Phương thức Abandon dùng để kết thúc session của người dùng. Khi phương thức này được gọi, đối tượng Session hiện hành chưa bị xóa ngay mà sẽ tồn tại cho tới khi tất cả các Script của trang hiện hành được xử lý xong. Điều này có nghĩa là chúng ta có thể truy cập các biến session trong cùng trang mặc dù chúng ta đã gọi phương thức Abandon trước đó, nhưng truy cập các biến session từ những trang khác thì không được.

Cú pháp:

Session.Abandon

- **Contents.Remove**

Phương thức này dùng để xóa một phần tử ra khỏi tập Contents của đối tượng Session.

Cú pháp:

Session.Contents.Remove (name | index)

Khi gọi phương thức này ta có thể truyền vào tên của phần tử cần xóa hoặc vị trí của phần tử trong tập Contents.

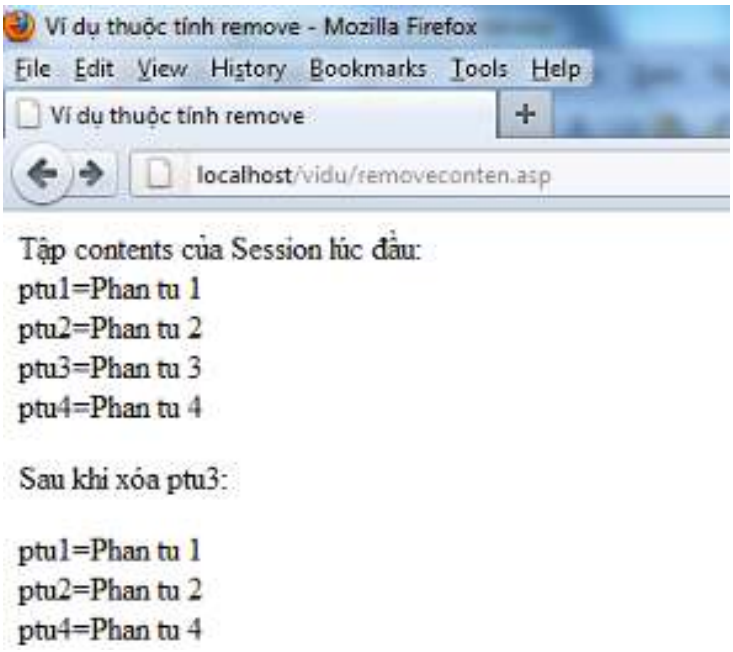
Ví dụ 4.33: Ví dụ này sẽ đọc ra các content rồi xóa bỏ một content rồi đọc lại các content đó.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Ví dụ thuộc tính remove</title>
</head>
<body>
   <%
   Session("ptu1") = "Phan tu 1"
   Session("ptu2") = "Phan tu 2"
   Session("ptu3") = "Phan tu 3"
   Session("ptu4") = "Phan tu 4"
   Response.Write ("Tập contents của Session lúc đầu: <br>")
   Dim x

   For each x in Session.Contents
      Response.Write (x & "=" & Session.Contents(x) & "<br>")
   next
   Session.Contents.Remove("ptu3")
   Response.Write ("<p> Sau khi xóa ptu3: </p>")
   For each x in Session.Contents
```

```
        Response.Write (x & "=" & Session.Contents(x) & "<br>")
    next
    %>
</body>
</html>
```

Kết quả như sau :



Hình 4.16. Kết quả chạy file removecontent.asp

- **Contents.RemoveAll()**

Thay vì chỉ xóa một phần tử ta dùng phương thức Remove thì phương thức này xóa tất cả các phần tử ra khỏi tập Contents.

Cú pháp:

Session.Contents.RemoveAll()

Các sự kiện của đối tượng Session

- **Session_OnStart**

Sự kiện này xuất hiện khi trình chủ tạo một session mới. Cài đặt của sự kiện này được đặt trong tập tin global.asa.

- Session_OnEnd

Sự kiện này xuất hiện khi session kết thúc. Cài đặt của sự kiện này cũng được đặt trong tập tin global.asa.

Chú ý: Trong cài đặt của sự kiện Session_OnEnd ta không sử dụng được phương thức MapPath bởi vì ở đây phương thức này không còn hiệu lực.

4.3.5. Đối tượng Application

Một ứng dụng bao gồm một tập hợp các file kết hợp với nhau để xử lý hoặc phục vụ cho một mục đích nào đó. ASP cung cấp một đối tượng dùng để kết hợp các file đó lại với nhau, đó là đối tượng Application.

Đối tượng Application được dùng để lưu trữ các biến, qua đó các trang có thể truy cập đến các biến này. Không giống như đối tượng Session chỉ dùng cho một nối kết cho mỗi người dùng, đối tượng Application được dùng chung cho tất cả các người dùng. Do đó đối tượng Application nên chứa các thông tin mà có thể được truy cập bởi nhiều trang trong ứng dụng (như thông tin nối kết cơ sở dữ liệu, thông tin về số người dùng truy cập, ...) nghĩa là chúng ta có thể truy cập các thông tin này từ bất cứ trang nào trong ứng dụng, nhưng chú ý là khi thay đổi các thông tin này sẽ ảnh hưởng đến tất cả các trang khác trong ứng dụng.

1) Tập hợp của đối tượng Application

- **Contents**

Tập hợp Contents chứa tất cả các phần tử đã được gắn thêm vào đối tượng Application trong quá trình thực thi script.

Cú pháp:

Application.Contents (Key)

Trong đó key là tên của phần tử cần lấy.

- **StaticObjects**

Tập hợp StaticObjects chứa tất cả các đối tượng được gắn vào ứng dụng với thẻ HTML <object>

Cú pháp:

Application.StaticObjects(Key)

Trong đó key là tên của phần tử cần lấy

2) Các phương thức của đối tượng Application

- **Contents.Remove**

Phương thức này dùng để xóa một phần tử ra khỏi tập Contents của đối tượng Application.

Cú pháp:

Application.Contents.Remove (name | index)

Khi gọi phương thức này ta có thể truyền vào tên của phần tử cần xóa hoặc vị trí của phần tử trong tập Contents.

- **Contents.RemoveAll**

Thay vì chỉ xóa một phần tử ta dùng phương thức Remove thì phương thức này xóa tất cả các phần tử ra khỏi tập Contents

Cú pháp:

Application.Contents.RemoveAll()

- **Lock và Unlock**

Bởi vì tất cả các người dùng đều có thể truy cập đến các biến Application nên có thể cùng lúc 2 hay nhiều người dùng cùng thay đổi giá trị của biến và điều này dẫn đến sai lệch giá trị của biến. Để khắc phục điều này đối tượng Application cung cấp hai phương thức Lock và Unlock. Phương thức Lock ngăn cản người dùng khác thay đổi biến trong đối tượng Application (dùng để đảm bảo rằng tại một thời điểm chỉ có một người dùng thay đổi các biến trong đối tượng Application). Phương thức Unlock cho phép người dùng thay đổi giá trị các biến trong đối tượng Application.

Cú pháp:

Application.Lock

Application.Unlock

Lưu ý: Khi gọi phương thức Lock thì ta phải nhớ gọi phương thức Unlock ngay khi thực hiện xong.

Ví dụ 4.34: đoạn chương trình sau đây sẽ thực hiện đếm số người thăm website.

```
<%
Application.Lock
Application("visits") = Application("visits") +1
Application.Unlock
%>
```

3) Các sự kiện (Events) của đối tượng Application

Cũng giống như đối tượng Session đôi tượng Application cũng có 2 sự kiện dùng để thiết lập các bên Application đó là:

- **Application_OnStart**

Sự kiện này xuất hiện trước khi một phiên nối kết mới đầu tiên được hình thành. Sự kiện này được đặt trong file global.asa

- **Application_OnEnd**

Sự kiện này xuất hiện khi ứng dụng kết thúc (khi web server dừng). Sự kiện này cũng được đặt trong file global.asa

4.3.6. Đối tượng Server

Đối tượng Server cung cấp nhiều thuộc tính và phương thức dùng để truy cập server. Đây là đối tượng dùng để quản lý những đặc trưng của trình chủ IIS và các hành động liên

quan tới dịch vụ HTTP. Ngoài ra đối tượng Server còn cung cấp khả năng tạo kế thừa các thành phần COM trên Server.

1) Các thuộc tính của đối tượng Server

Đối tượng Server chỉ có duy nhất một thuộc tính đó là ScriptTimeOut. Thuộc tính này quy định thời gian lớn nhất mà các lệnh kịch bản còn được thực hiện. Giá trị mặc định là 90 giây.

Lưu ý: Là giá trị timeout sẽ không hiệu lực khi server thực hiện các lệnh kịch bản.

Cú pháp:

Server.ScriptTimeOut = [number]

2) Các phương thức của đối tượng Server

- **CreateObject**

Phương thức CreateObject dùng để tạo một thực thể của một đối tượng. Các đối tượng do phương thức này tạo ra chỉ có hiệu lực trong phạm vi một trang, do đó chúng sẽ bị hủy khi server xử lý xong trang ASP hiện hành.

Để tạo một đối tượng mà phạm vi của nó như Session hay Application, chúng ta có thể dùng tag <object> trong file Global.asa hoặc lưu trữ đối tượng trong biến Session hay Application.

Cú pháp:

Server.CreateObject (progID)

Trong đó progID là kiểu của đối tượng cần tạo.

- **Execute**

Thuộc tính Execute thực thi một trang ASP bên trong một trang khác. Sau khi thực thi xong file ASP được gọi thì quyền điều khiển được trả về cho file ASP ban đầu (file gọi).

Cú pháp:

Server.Execute (path)

Với path là đường dẫn tới tập tin ASP cần thực thi.

Ví dụ 4.35: Tạo ra 2 file sau đây trong cùng thư mục:

File1.asp

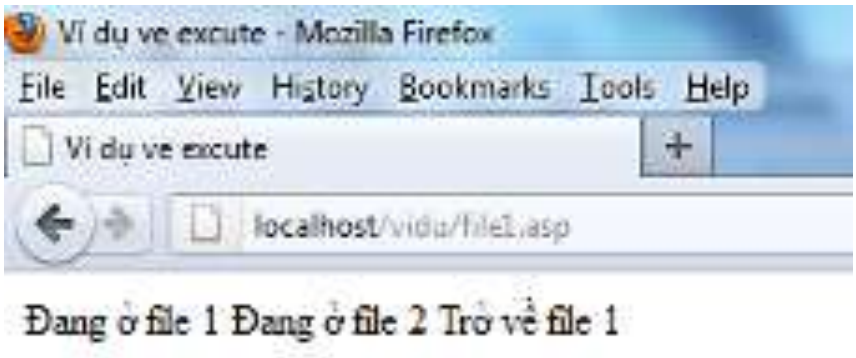
```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Ví dụ ve excute</title>
</head>
<body>
```

```
<%
    Response.Write "Đang ở file 1"
    Server.Execute ("File2.asp")
    Response.Write " Trở về file 1"
%>
</body>
</html>
```

File2.asp

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>File 2</title>
</head>
<body>
<%
    Response.Write "Đang ở file 2"
%>
</body>
</html>
```

Khi chạy chương trình file1.asp sẽ tự động gọi thực hiện file2.asp sau khi thực hiện xong nó sẽ trở về file1 kết quả thu được như sau:



Hình 4.17. Kết quả chạy file file1.asp

- **GetLastError**

Phương thức này trả về một đối tượng ASPError mô tả lỗi xuất hiện. Mặc định trang web dùng tập tin \iishelp\common\500-100.asp để xử lý các lỗi trong ASP. Nếu cần thì chúng ta có thể tạo hoặc thay đổi tập tin để đưa ra những câu thông báo thân thiện hơn,..

Chú ý: Phương thức này được dùng trước khi tập tin ASP gởi bất cứ nội dung gì xuống bowser.

Cú pháp:

Server.GetLastError()

- **HTMLEncode**

Phương thức này dùng để mã hóa dạng HTML thành chuỗi với cú pháp như sau:

Cú pháp:

Server.HTMLEncode(string)

Khi mã hóa chuỗi HTML sang chuỗi bình thường nếu gặp các ký tự như sau sẽ chuyển thành các ký tự tương ứng

- Ký tự < chuyển thành <
- Ký tự > chuyển thành >
- Ký tự & chuyển thành &
- Dấu nháy đôi " chuyển thành "

- **MapPath**

Phương thức này ánh xạ một đường dẫn ảo thành một đường dẫn vật lý cho một tập tin trên server. Phương thức này không được dùng trong sự kiện Session_OnEnd và Application_OnEnd.

Cú pháp:

Server.MapPath (path)

Trong đó tham số path là chuỗi hiện thực đường dẫn tương đối của tập tin trên ứng dụng web.

Chú ý: Nếu path bắt đầu bằng ký tự / hoặc \ thì các ký tự này đại diện cho đường dẫn vật lý của thư mục ảo của tập tin ASP hiện tại.

Ví dụ 4.36: sau đây sẽ đọc ra đường dẫn của file hiện hành:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>ví dụ về MapPath</title>
</head>
<body>
   <%
      Response.Write(Server.MapPath("test.asp") & "<br>")
      Response.Write(Server.MapPath("/") & "<br>")
      Response.Write(Server.MapPath("\") & "<br>")
   %>
</body>
</html>
```

Kết quả thực hiện chương trình như sau:

Hình 4.18. Kết quả chạy file Test.asp

- **Transfer**

Phương thức này gởi (chuyển) tất cả các thông tin về trạng thái (các biến Session, các biến Application, các dữ liệu trong tập Request...) của tập tin ASP hiện tại cho một tập tin ASP khác. Khi trang thứ hai thực hiện xong thì quyền điều khiển không trả về cho trang trước đó (xem thêm phương thức Execute). Phương thức Transfer là một dạng khác của phương thức Response.Redirect nhưng lại hiệu quả hơn bởi vì phương thức Response.Redirect buộc Server phải giữ lại một Request giả trong khi phương thức Server.Transfer thì chuyển quyền điều khiển cho một trang ASP khác trên server. (xem thêm phương thức Response.Redirect).

Cú pháp:

Server.Transfer (path)

Trong đó tham số path là đường dẫn của tập tin asp cần chuyển điều khiển.

Ví dụ 4.37:

Ta tạo ra 2 file, file1.asp sẽ transfer sang file2.asp như sau:

Nội dung File1.asp

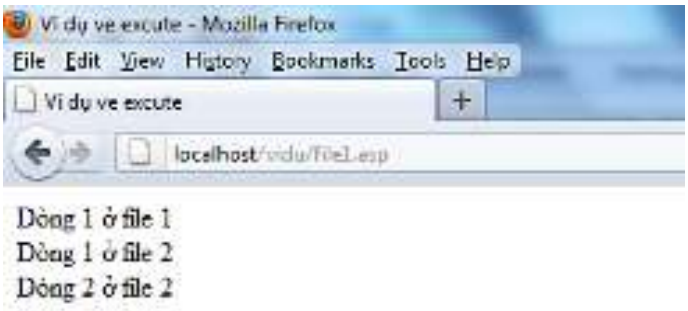
```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Ví dụ ve excute</title>
</head>
<body>
   <%
      Response.Write "Dòng 1 ở file 1 <br>"
      Server.Transfer("File2.asp")
      Response.Write " Dòng 2 ở file 1"
   %>
</body>
```

```
</html>
```

Nội dung file2.asp

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>File 2</title>
</head>
<body>
   <%
        Response.Write "Dòng 1 ở file 2 <br>"
        Response.Write "Dòng 2 ở file 2"
   %>
</body>
</html>
```

Kết quả khi thục thi file1.asp ta được như sau:



Hình 4.19. Kết quả chạy file file1.asp

- **URLEncode**

Tương tự như phương thức HTMLEncode, phương thức này dùng để mã hóa một URL thành chuỗi.

Cú pháp:

Server.URLEncode (stringURL)

Khi mã hóa chuỗi url thành chuỗi bình thường nếu gặp các ký tự như sau, sẽ chuyển sang ký tự tương ứng.

- Ký tự khoảng trắng chuyển thành dấu +
- Ký tự không thuộc ký tự chữ và số sẽ chuyển thành số hexadecimal.

4.3.7. Đối tượng ASPError

Đối tượng ASPError được dùng để hiển thị thông tin chi tiết của bất cứ lỗi nào xuất hiện trong các kịch bản của trang ASP. Đối tượng ASPError được tạo ra khi phương thức

Server.GetLastError được gọi, vì thế thông tin về các lỗi chỉ có thể được truy cập bằng việc gọi phương thức Server.GetLastError. Đối tượng ASPError được bổ sung vào ASP từ phiên bản 3.0 trở đi và chỉ có sẵn trong IIS 5.

Đối tượng ASPError không có phương thức nào mà chỉ có các thuộc tính để cung cấp các thông tin về lỗi xuất hiện.

Các thuộc tính của đối tượng ASPError

- **ASPCode**

Thuộc tính này cho biết mã lỗi được tạo ra bởi IIS, chúng ta có thể sử dụng thuộc tính Aspcode để nhận mã lỗi.

Cú pháp:

ASPError.ASPCode

- **ASPDescription**

Thuộc tính này trả về một chuỗi mô tả chi tiết lỗi nội dung phát sinh ra lỗi.

Cú pháp:

ASPError.ASPDescription

- **Category**

Thuộc tính này cho biết nơi nào đưa ra lỗi (do IIS hay do ngôn ngữ kịch bản hay do một thành phần phụ thêm nào đó).

Cú pháp:

ASPError.Category

- **Column**

Thuộc tính này cho biết vị trí cột thứ mấy trong tập tin ASP đã gây ra lỗi.

Cú pháp:

ASPError.Column

- **Description**

Thuộc tính này mô tả nội dung tóm tắt về lỗi được phát sinh.

Cú pháp:

ASPError.Description

- **File**

Thuộc tính này trả về tên tập tin ASP đã gây ra lỗi.

Cú pháp:

ASPError.File

- **Line**

Thuộc tính này cho biết dòng thứ mấy trong tập tin ASP đã gây ra lỗi.

Cú pháp:

ASPError.Line

- **Number**

Thuộc tính này trả về mã lỗi chuẩn dạng số tương ứng với số nhận dạng COM chuẩn của lỗi tạo ra.

Cú pháp:

ASPError.Number

- **Source**

Thuộc tính này trả về đoạn mã của dòng gây ra lỗi.

Cú pháp:

ASPError.Source

Ví dụ 4.38:

đoạn chương trình sau đây sẽ đọc ra tất cả các thông tin liên quan đến lỗi phát sinh:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Phát sinh lỗi</title>
</head>
<body>
   <%
   Dim i
   Dim j
   Dim k
   i=0
   j=10
   'k =j/i
   Dim objErr
   Set objErr = Server.GetLastError()
   Response.Write ("ASPCode = " & objErr.ASPCode)
   Response.Write ("<br>")
   Response.Write("ASPDescription= "&objErr. ASPDescription)
   esponse.Write ("<BR>")
   Response.Write ("Category = " & objErr.Category)
   Response.Write ("<BR>")
   Response.Write ("Column = " & objErr.Column)
   Response.Write ("<BR>")
   Response.Write ("Description = " & objErr. Description)
   Response.Write ("<BR>")
```

```
    Response.Write ("File = " & objErr.File)
    Response.Write ("<BR>")
    Response.Write ("Line = " & objErr.Line)
    Response.Write ("<BR>")
    Response.Write ("Number = " & objErr.Number)
    Response.Write ("<BR>")
    Response.Write ("Source = " & objErr.Source)
    Response.Write ("<BR>")
    %>
</body>
</html>
```

4.4. Tập tin Global.asa

ASP cung cấp một file cấu hình global.asa, trong file này chúng ta có thể đặt các script xử lý các sự kiện hay các hàm, thủ tục, biến mang tính toàn cục. File global.asa phải được đặt trong thư mục gốc của ứng dụng và mỗi ứng dụng chỉ được phép có duy nhất một file global.asa. Khi trang asp của ứng dụng được triệu gọi lần đầu tiên, trình chủ IIS sẽ tìm xem trong thư mục hiện tại của ứng dụng có file global.asa không. Nếu có thì trình chủ sẽ nạp và xử lý các sự kiện được cài đặt trong file này, sau đó chuyển giao quyền xử lý lại cho trang ASP. Trong file global.asa, chúng ta chỉ được phép cài đặt và xử lý các sự kiện sau:

Application_OnStart: Sự kiện này được phát sinh khi người dùng đầu tiên triệu gọi bất kỳ trang nào trong ứng dụng. Khi trình chủ IIS khởi động lại hoặc khi nội dung file global.asa bị hiệu chỉnh thì sự kiện này được phát sinh trở lại. Sau khi xử lý xong sự kiện này, trình chủ bắt đầu xử lý sự kiện Session_OnStart để chuẩn bị cho phiên nối kết. Các biến Application thường được khởi tạo bên trong sự kiện này.

Session_OnStart: Sự kiện này được gọi mỗi khi có một người dùng mới yêu cầu trang asp của ứng dụng Web trong lần đầu tiên. Các biến session của người dùng cũng thường được khởi tạo bên trong sự kiện này.

Session_OnEnd: Sự kiện này được gọi khi phiên làm việc của người dùng chấm dứt. Phiên làm việc được xem là chấm dứt khi nó hết hạn (timeout hay expired), mặc định cho thời gian làm việc của session là 20 phút, chúng ta có thể tăng hay giảm thời gian này bằng cách thay đổi giá trị của thuộc tính Timeout của đối tượng session.

Application_OnEnd: Sự kiện này được gọi khi không còn người dùng nào tương tác với ứng dụng web của chúng ta nữa. Thông thường thì sự kiện này được gọi khi trình chủ IIS ngừng hoạt động. Thông qua sự kiện này chúng ta có thể giải phóng vùng nhớ đã cấp phát trước đó hoặc lưu lại các thông tin, trạng thái cần thiết xuống đĩa cứng để phục vụ cho quá trình khởi động trở lại sau đó.

Chúng ta có thể cài đặt thủ tục xử lý sự kiện trong file global.asa theo mẫu sau:

```
<script language = "vbscript" runat = "server">
  Sub Application_OnStart
    ' đoạn mã cho sự kiện application
  End Sub
  Sub Session_OnStart
    ' đoạn mã cho sự kiện session
  End Sub
  Sub Session_OnEnd
    ' đoạn mã cho sự kiện kết thúc session
  End Sub
  Sub Application_OnEnd
    ' đoạn mã cho sự kiện kết thúc application
  End Sub
  </script>
```

Ví dụ 4.39: Ta có tệp tin Global.asa như sau để thiết lập các biến Application và Session.

```
<Script Language = "VBScript" RunAt = Server>
Sub Application_OnStar()
  Application("NumSession")=0
  Application("NumVisited")=0
End Sub
Sub Application_OnEnd()
  Calculate_Stats()
End Sub
Sub Session_OnStar()
  Application("NumSession") = Application("NumSession") + 1
  Application("NumVisited") = Application("NumVisited") + 1
  ' Set objCon=Server.CreateObject("ADODB.Connection")
  ' Câu lệnh trên có thể đưa vào Application_OnStart()
End Sub
Sub Session_OnEnd()
  Application("NumSession") = Application("NumSession") - 1
End Sub
</Script>
```

4.5. Đối tượng Dictionary

Đối tượng Dictionary được dùng để lưu trữ thông tin theo cặp tên/giá trị. Đối tượng dictionary có thể xem tương tự như mảng, tuy nhiên đối tượng Dictionary được tạo ra để thao tác với dữ liệu một cách hiệu quả hơn.

So sánh đối tượng Dictionary với các mảng ta thấy:

+ Đối tượng Dictionary dùng từ khoá (key) để nhận diện các phần tử (item) còn mảng thì sử dụng chỉ số.

+ Chúng ta không thể dùng ReDim để thay đổi kích thước của đối tượng Dictionary còn mảng thì được.

+ Khi xoá một phần tử khỏi đối tượng Dictionary thì các phần tử còn lại sẽ tự động thay thế, còn các mảng thì không.

+ Mảng có thể có nhiều chiều còn đối tượng Dictionary thì không.

+ Đối tượng Dictionary được xây dựng với nhiều chức năng hơn.

+ Đối tượng Dictionary truy cập thường xuyên các phần tử một cách ngẫu nhiên hiệu quả hơn mảng.

+ Đối tượng Dictionary định vị các phần tử dựa trên nội dung hiệu quả hơn.

4.5.1. Tạo các đối tượng Dictionary

Đối tượng Dictionary được tạo ra bởi đối tượng Server bằng việc gọi phương thức CreateObject như sau:

```
<%
    Dim Dic
    Set Dic = Server.CreateObject("Scripting.Dictionary")
%>
```

Bởi vì hàm CreateObject của đối tượng Server trả về một đối tượng nên để gán đối tượng cho biến Dic ta dùng lệnh Set.

Khi sử dụng xong thực thể của đối tượng Dictionary ta phải hủy bỏ thực thể đó bằng cách:

```
Set Dic = nothing
```

4.5.2. Các thuộc tính của đối tượng Dictionary

- **CompareMode**

Ta dùng thuộc tính **CompareMode** để thiết lập hoặc nhận về chế độ so sánh để so sánh các khoá trong đối tượng **Dictionary**.

Cú pháp:

```
Dictionary.CompareMode [=mode]
```

Trong đó mode có thể nhận một trong các giá trị sau:

0 = vbBinaryCompare – So sánh nhị phân

1 = vbTextCompare – So sánh dạng văn bản

2 = vbDatabaseCompare – So sánh cơ sở dữ liệu

- **Count**

Thuộc tính này trả về số cặp tên/giá trị (số phần tử) trong đối tượng Dictionary.

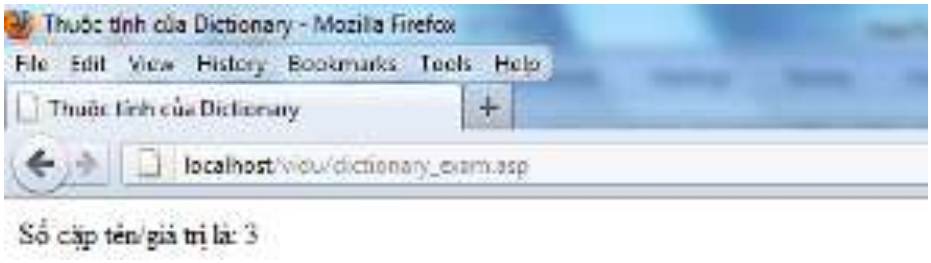
Cú pháp:

Dictionary.Count

Ví dụ 4.40: sau đây tạo ra môt Dictionary và sử dụng thuộc tính count để đếm số phần tử:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Thuộc tính của Dictionary</title>
</head>
<body>
  <%
    Dim Dic
    Set Dic = Server.CreateObject("Scripting.Dictionary")
    Dic.CompareMode = 1
    Dic.Add "HN", "Hà Nội"
    Dic.Add "HCM", "Hồ Chí Minh"
    Dic.Add "HP", "Hải Phòng"
    Response.Write ("Số cặp tên/giá trị là: " & Dic.Count)
    Set Dic = nothing
  %>
</body>
</html>
```

Kết quả của ví dụ là:



Hình 4.20. Kết quả chạy file dictionary_exam.asp

- **Item**

Dùng thuộc tính này để gán hoặc lấy về giá trị của một phần tử trong đối tượng Dictionary.

Cú pháp:

Dictionary.Item (key)[= newitem]

Ví dụ 4.41: để lấy thông tin về phần tử HN trong đối tượng Dictionary trong ví dụ trên ta làm như sau:

```
html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Thuộc tính của Dictionary</title>
</head>
<body>
   <%
      Dim Dic
      Set Dic = Server.CreateObject("Scripting.Dictionary")
      Dic.CompareMode = 1
      Dic.Add "HN", "Hà Nội"
      Dic.Add "HCM", "Hồ Chí Minh"
      Dic.Add "HP", "Hải Phòng"
      Response.Write ("Số cặp tên/giá trị là: " & Dic.Count)
        Response.Write ("Giá trị của khoá HN là: " & Dic.Item("HN"))
      Set Dic = nothing
   %>
</body>
</html>
```

- **Key**

Để thay đổi tên của một khoá đã có trong đối tượng Dictionary ta dùng thuộc tính key theo cú pháp sau:

Dictionary.Key (key)[= newkey]

4.5.3. Các phương thức của đối tượng Dictionary

- **Add**

Phương thức Add dùng để thêm một cặp khoá/giá trị mới vào đối tượng Dictionary. Nếu khoá này đã có trong Dictionary thì phương thức này sẽ bị sai.

Cú pháp:

Dictionary.Add (key, value)

- **Exists**

Để kiểm tra một khoá đã có trong đối tượng Dictionary hay chưa ta dùng phương thức Exists. Phương thức này trả về true nếu khoá đã có trong Dictionary và trả về false nếu khoá này tồn tại.

Cú pháp:

Dictionary.Exists (key, value)

Ví dụ 4.42:

Kiểm tra xem một khóa có tồn tại hay không bằng sử dụng thuộc tính Exist.

```
<HTML>
<BODY>
<%
Dim Dic
Set Dic = Server.CreateObject("Scripting.Dictionary")
Dic.Add "Đ", "Đỏ"
Dic.Add "X", "Xanh"
Dic.Add "V", "Vàng"
Dic.Key("T") = "Tím"
If Dic.Exists ("V") = true Then
  Response.Write "Khoá V tồn tại!"
Else
  Response.Write "Khoá V không tồn tại!"
End If
Set Dic = nothing
%>
</BODY>
</HTML>
```

- **Items**

Không phải lúc nào ta cũng thao tác trên các khoá của đối tượng Dictionary mà đôi lúc ta cũng phải thao tác trên dữ liệu của các khoá như: tìm kiếm một giá trị nào đó, sửa đổi giá trị,... Nếu ta duyệt lần lượt trên các khoá và lấy giá trị của chúng để so sánh thì ắt hẳn sẽ mất nhiều thời gian.

Đối tượng Dictionary cung cấp cho ta phương thức Items để lấy một mảng các giá trị của các khoá, và nhờ vào mảng này ta sẽ thao tác trên dữ liệu dễ dàng hơn. Cú pháp của phương thức Items như sau:

Dictionary.Items

Ví dụ 4.43:

Đọc ra tất cả các phần tử có trong một Dictionary:

```
<HTML>
<BODY>
<%
Dim Dic
Set Dic = Server.CreateObject("Scripting.Dictionary")
Dic.Add "Đ", "Đỏ"
Dic.Add "X", "Xanh"
Dic.Add "V", "Vàng"
Dic.Key("T") = "Tím"
  Response.Write("<p>" & "Các giá trị: " & "</p>")
Dim Arr, i
Arr = Dic.Items
For i = 0 to Dic.Count-1
  Response.Write (Arr(i) & "<br>")
Next
Set Dic = nothing
%>
</BODY>
</HTML>
```

- **Keys**

Thay vì trả về một mảng các giá trị như phương thức Items thì phương thức Keys lại trả về một mảng các khoá trong đối tượng Dictionary.

Cú pháp:

Dictionary.Keys

- **Remove**

Phương thức này xoá một phần tử (một cặp khoá/giá trị) ra khỏi đối tượng Dictionary. Cú pháp của phương thức này như sau:

Dictionary.Remove (key)

Ví dụ 4.44:

Hủy bỏ một phần tử có khóa D trong Dictionary ở trên:

```
<HTML>
<BODY>
<%
Dim Dic
Set Dic = Server.CreateObject("Scripting.Dictionary")
Dic.Add "Đ", "Đỏ"
```

```
Dic.Add “X”, “Xanh”
Dic.Add “V”, “Vàng”
Dic.Key(“T”) = “Tím”
  Response.Write(“<p>” & “Các giá trị trước khi gọi Remove: ” & “</p>”)
Dim Arr, i
Arr = Dic.Items
For i = 0 to Dic.Count-1
  Response.Write (Arr(i) & “<br>”)
Next
Dic.Remove(“Đ”)
  Response.Write(“<p>” & “Các giá trị sau khi gọi Remove: ” & “</p>”)
Dim Arr, i
Arr = Dic.Items
For i = 0 to Dic.Count-1
  Response.Write (Arr(i) & “<br>”)
Next
Set Dic = nothing
%>
</BODY>
</HTML>
```

- **RemoveAll**

Phương thức này dùng để xoá tất cả các phần tử của đối tượng Dictionary.

Cú pháp:

Dictionary.RemoveAll

Ví dụ 4.45:

```
<HTML>
<BODY>
<%
Dim Dic
Set Dic = Server.CreateObject(“Scripting.Dictionary”)
Dic.Add “Đ”, “Đỏ”
Dic.Add “X”, “Xanh”
Dic.Add “V”, “Vàng”
Dic.Key(“T”) = “Tím”
  Response.Write(“<p>” & “Các giá trị trước khi gọi RemoveALL: ” & “</p>”)
Dim Arr, i
```

```
Arr = Dic.Items
For i = 0 to Dic.Count-1
  Response.Write (Arr(i) & "<br>")
Next
Dic.RemoveAll
Response.Write("<p>" & "Số phần tử có trong dictionary là:" &Dic.count&
"</p>")
Set Dic = nothing
%>
</BODY>
</HTML>
```

4.6. Đối tượng FileSystemObject

Đối tượng FileSystemObject cung cấp thông tin về hệ thống tập tin, thư mục trên trình chủ và ta có thể sử dung đối tượng này để thao tác với các tập tin, thư mục,...

4.6.1. Tạo các đối tượng FileSystemObject

Bởi vì đối tượng FileSystemObject thao tác trên hệ thống tập tin của trình chủ (server) nên đối tượng này đối tượng này được tạo ra bởi Server theo cú pháp sau:

```
<%
  Dim fso
  Set fso = Server.CreateObject("Scripting.FileSystemObject")
%>
```

Lưu ý: Khi dùng xong thực thể của đối tượng FileSystemObject ta phải hủy thực thể đó đi bằng cách: Set fso = nothing

4.6.2. Các thuộc tính của đối tượng FileSystemObject

Đối tượng FileSystemObject chỉ có một thuộc tính duy nhất đó là: Drives:Thuộc tính này cho biết một tập tất cả các ổ dĩa trên máy tính.

Cú pháp:

[drivecoll =] FileSystemObject.Drives

4.6.3. Các phương thức của đối tượng FileSystemObject

- **BuildPath**

Phương thức này gắn một chuỗi vào một đường dẫn đã có để tạo ra một đường dẫn mới.

Cú pháp:

[newpath =]FileSystemObject.BuildPath(path, name)

Trong đó:

+ path: là đường dẫn đã tồn tại

+ name: là tên cần gắn thêm vào Path

+ newpath: là đường dẫn mới sau khi đã gắn tên vào

Ví dụ 4.46: Tạo ra một đường dẫn mới:

```
<html>
<body>
<%
Dim fso, NewPath
Set fso = Server.CreateObject("Scripting.FileSystemObject")
NewPath = fso.BuildPath("C:\My Documents", "BT")
Response.Write (NewPath)
Set fso = nothing
%>
</body>
</html>
```

Trong ví dụ trên, sau khi gọi phương thức BuildPath thì biến NewPath sẽ có giá trị là "C:\My Documents\BT"

- **CopyFile**

Phương thức này sao chép một hoặc một số tập tin từ thư mục này tới thư mục khác.

Cú pháp:

FileSystemObject.Copy src, des [,ovr]

Trong đó:

src: là đường dẫn tới tập tin cần sao chép, tên tập tin cần chép có thể chứa các ký tự thay thế như (*, ?).

des: Là đường dẫn của thư mục cần chép tới, đường dẫn này không được chứa ký tự thay thế (*, ?).

ovr: nhận giá trị true hoặc false. Nếu ovr là true có nghĩa là cho phép chép đè lên các tập tin đã có trong des. Nếu false thì không cho chép đè. Giá trị mặc định của ovr là true.

Ví dụ 4.47: Đoạn chương trình sau đây sao chép tất cả các tập tin có đuôi .asp trong thư mục C:\Web sang thư mục D:\ASP

```
<HTML>
<BODY>
<%
Dim fso
Set fso = Server.CreateObject("Scripting.FileSystemObject")
```

```
fso.Copy "C:\Web\*.asp", "D:\ASP"
Set fso = nothing
%>
</BODY>
</HTML>
```

- **CopyFolder**

Phương thức này sao chép một hoặc nhiều thư mục.

Cú pháp:

FileSystemObject.CopyFolder src, des [,ovr]

Ví dụ 4.48: Sao chép tất cả các thư mục con của thư mục C:\Web vào thư mục D:\ASP

```
<HTML>
<BODY>
<%
Dim fso
Set fso = Server.CreateObject("Scripting.FileSystemObject")
fso.CopyFolder "C:\Web\*.asp", "D:\ASP"
Set fso = nothing
%>
</BODY>
</HTML>
```

- **CreateFolder**

Phương thức này tạo một thư mục mới.

Cú pháp:

FileSystemObject.CreateFolder (Foldername)

Ví dụ 4.49: Tạo thư mục C:\ASP

```
<HTML>
<BODY>
<%
Dim fso
Set fso = Server.CreateObject("Scripting.FileSystemObject")
Fso.CreateFolder "C:\ASP"
Set fso = nothing
%>
</BODY>
</HTML>
```

- **CreateTextFile**

Phương thức này tạo một tập tin văn bản trong thư mục hiện hành và trả về một đối tượng TextStream dùng để đọc hoặc ghi dữ liệu lên file.

Cú pháp:

FileSystemObject.CreateTextFile(filename[,Ovr[,Uni]])

- **DeleteFile**

Phương thức này xoá một hoặc nhiều tập tin. Nếu tập tin không tồn tại thì sẽ xuất hiện lỗi.

Cú pháp:

FileSystemObject.DeleteFile(filename[,bReadOnly])

Trong đó bReadOnly nhận một trong hai giá trị. Nếu nhận giá trị true thì các tập tin mang thuộc tính chỉ đọc (read-only) cũng sẽ bị xoá. Nếu nhận giá trị false thì các tập tin mang thuộc tính read-only sẽ không bị xoá.

- **DeleteFolder**

Phương thức này xoá một hoặc nhiều thư mục. Nếu thư mục không tồn tại thì phương thức này sẽ gây ra lỗi.

Cú pháp:

FileSystemObject.DeleteFolder(foldername[,bReadOnly])

- **DriveExists**

Phương thức DriveExists kiểm tra trên hệ thống tập tin của server có tồn tại một ổ dĩa nào đó hay không? Nếu có thì phương thức này trả về true, còn nếu không thì sẽ trả về false.

Cú pháp:

FileSystemObject.DriveExists(drive)

Trong đó drive là tên của ổ dĩa cần kiểm tra

Ví dụ 4.50: kiểm tra ổ đĩa c có tồn tại không.

```
<HTML>
<BODY>
<%
Dim fso
Set fso = Server.CreateObject("Scripting.FileSystemObject")
if fso.DriveExists("C:") = true then
  Response.Write ("Ổ dĩa C tồn tại!")
else
  Response.Write ("Ổ dĩa C không tồn tại!")
end if
Set fso = nothing
```

%>

</BODY>

</HTML>

- **GetAbsolutePathName**

Phương thức này trả về đường dẫn dạng đầy đủ của một đường dẫn tương đối.

Cú pháp:

FileSystemObject.GetAbsolutePathName(path)

Ví dụ 4.51: Giả sử đường dẫn hiện hành là C:\ASP. Đoạn chương trình sau đây sẽ in ra màn hình browser dòng.

C:\ASP\Data\list.txt

```
<HTML>
<BODY>
  <%
   Dim fso, path
   Set fso = Server.CreateObject("Scripting.FileSystemObject")
   path = fso.GetAbsolutePathName("Data\list.txt")
   Response.Write (path)
   Set fso = nothing
   %>
</BODY>
</HTML>
```

- **GetBaseName**

Phương thức này trả về phần tên của một tập tin hoặc tên của thư mục ở cuối một đường dẫn.

Cú pháp:

FileSystemObject.GetBaseName(path)

Ví dụ 4.52: Nếu path = "C:\ASP\Data\list.txt" thì hàm này sẽ trả về "list"

- **GetDrive**

Phương thức này trả về một đối tượng Drive mô tả một ổ đĩa. Có được đối tượng Drive ta có thể thao tác trên ổ đĩa mà Drive mô tả bằng cách sử dụng các phương thức của đối tương Drive.

Cú pháp:

FileSystemObject.GetDrive(Drive)

- **GetDriveName**

Phương thức này trả về một chuỗi là tên của ổ đĩa trong một đường dẫn.

Cú pháp:

FileSystemObject.GetDriveName(path)

Ví dụ 4.53: Nếu path = "C:\ASP\Data\list.txt" khi gọi hàm GetDriveName(path) ta sẽ nhận được chuỗi "C:"

- **GetExtensionName**

Phương thức này trả về phần mở rộng của một tập tin (không bao gồm dấu chấm phân cách giữa phần tên và phần mở rộng).

Cú pháp:

FileSystemObject.GetExtensionName(path)

Ví dụ 4.54:

Nếu path = "C:\ASP\Data\list.txt" thì khi gọi hàm GetExtensionName(path) ta sẽ nhận được chuỗi "txt"

- **GetFile**

Phương thức GetFile trả về đối tượng File mô tả một tập tin đã được chỉ định trong đường dẫn truyền vào.

Cú pháp:

FileSystemObject.GetFile(path)

- **GetFileName**

Phương thức này chỉ trả về phần tên của một tập tin hay một thư mục.

Cú pháp:

FileSystemObject.GetFileName(path)

Ví dụ 4.55: Nếu path = "C:\ASP\Data" thì khi gọi hàm GetFileName(path) ta sẽ nhận được chuỗi "Data". Nếu path = "C:\ASP\Data\list.txt" thì khi gọi hàm GetFileName(path) ta sẽ nhận được chuỗi "list".

- **GetFolder**

Phương thức GetFolder trả về đối tượng Folder của một thư mục.

Cú pháp:

FileSystemObject.GetFolder(path)

- **GetParentFolderName**

Phương thức này trả về thư mục cha của một thư mục.

Cú pháp:

FileSystemObject.GetParentFolderName(path)

Ví dụ 56: Nếu path = "C:\ASP\Data" thì khi gọi hàm GetParentFolderName(path) ta sẽ nhận được chuỗi ASP". Đây là thư mục cha của thư mục Data

- **GetSpecialFolder**

Phương thức này trả về đường dẫn tới một số thư mục đặc biệt của hệ điều hành.

Cú pháp:

FileSystemObject.GetSpecialFolder(foldername)

Trong đó: foldername nhận một trong các giá trị sau:

+ WindowsFolder hay 0: Nếu muốn nhận về thư mục của hệ điều hành (mặc định Windows 98 đó là thư mục

Windows, đối với Windows 2000 thì đó là thư mục Winnt).

+ SystemFolder hay 1: Nếu muốn nhận về đường dẫn tới thư mục System của hệ điều hành.

+ TemporaryFolder hay 2: Nếu muốn nhận về đường dẫn tới thư mục tạm thời (TEMP) của hệ điều hành.

Ví dụ 4.57: Đoạn chương trình sau đây lấy thư mục hệ thống của hệ điều hành. Nếu dùng Windows 2000 thì trên màn hình browser sẽ xuất hiện dòng "C:\WINNT\System32"

```
<HTML>
<BODY>
    <%
      Dim fso, path
      Set fso = Server.CreateObject("Scripting.FileSystemObject")
      path = fso.GetSpecialFolder(1)
      Response.Write (path)
      Set fso = nothing
    %>
</BODY>
</HTML>
```

- **GetTempName**

Phương thức này trả về một tên tập tin hoặc thư mục tạm thời được phát sinh ngẫu nhiên.

Cú pháp:

FileSystemObject.GetTempName

- **MoveFile**

Phương thức này di chuyển một hoặc nhiều tập tin từ nơi này sang nơi khác.

Cú pháp:

FileSystemObject.MoveFile (src, des)

Trong đó src là nơi chứa các tập tin cần di chuyển di, des là nơi mà các tập tin cần chép đến.

- **MoveFolder**

Phương thức này di chuyển một hoặc nhiều thư mục từ nơi này sang nơi khác.

Cú pháp:

FileSystemObject.MoveFile (src, des)

Trong đó src là nơi chứa các tập tin cần di chuyển di, des là nơi mà các tập tin cần chép đến.

- **OpenTextFile**

Phương thức này mở một tập tin và trả về một đối tượng TextStream được dùng để truy cập đối tượng này.

Cú pháp:

FileSystemObject.OpenTextFile(fname, mode, creat, format)

Trong đó:

+ fname: là tên của tập tin cần mở

+ mode: dùng để chỉ cách thức mở.

+ create: dùng để chỉ định rằng nếu tập tin không tồn tại thì có tạo tập tin mới hay không.

+ format: dùng để chỉ ra rằng mở tập tin dùng chuẩn ASCII hay Unicode

4.7. Đối tượng AdRotator

Đối tượng AdRotator cho phép trình bầy hình ảnh quản cáo trên ứng dụng web một cách ngẫu nhiên khi người dùng truy nhập vào trang ASP.

Do không gian dành cho hình ảnh quảng cáo trên phần đầu của trang ASP có giới hạn mà số lượng cành nhiều hình ảnh xuất hiện, chính vì vậy chúng ta cần dùng đối tượng này để luân phiên thay đổi hình ảnh trong cùng không gian sao cho hợp lý.

4.7.1. Tạo các đối tượng AdRotator

Đối tượng AdRotator được tạo ra bởi đối tượng Server bằng việc gọi phương thức CreateObject như sau:

```
<%
    Dim ad
    Set ad=Server.CreateObject("MSWC.Adrotator")
%>
```

Bởi vì hàm CreateObject của đối tượng Server trả về một đối tượng nên để gán đối tượng cho biến ad ta dùng lệnh Set.

Đối tượng AdRotator sử dụng các thông tin được đọc trong một tập tin dạng Text có tên nên để thực hiện làm việc được với AdRotator ta cần phải tạo tập tin dạng Text có khuôn dạng như sau:

[REDIRECT URL]

[WIDTH numWidth]

[HEIGHT numHeight]

[BORDER numBorder]

*

adURL

adHomePageURL

Text

Impressions

Trong đó:

REDIRECT URL –Là đường dẫn đến tập tin asp sẽ hiển thị AdRotator

WIDTH numWidth - Khai báo chiều rộng của ảnh

HEIGHT numHeight - Khai báo chiều cao của ảnh

BORDER numBorder – Khai báo đường viền

Dấu * dùng để phần cách giữa các phần

adURL – Đường dẫn đến file ảnh

adHomePageURL – địa chỉ website

Text – Dòng văn bản

Impressions – độ trễ

Ví dụ: tạo ads.txt như sau

REDIRECT banners.asp

WIDTH 468

HEIGHT 60

BORDER 0

*

w3s.gif

...

4.7.2. Các thuộc tính của đối tượng AdRotator

- **Border**

Thuộc tính quy định cỡ của đường viền bao quanh ảnh làm quảng cáo

Cú pháp như sau:

AdRotator.Border = n

Trong đó n là một số là cỡ của đường viền

Ví dụ 4.58:

```
<%
    set adrot=Server.CreateObject("MSWC.AdRotator")
    adrot.Border="2"
    Response.Write(adrot.GetAdvertisement("ads.txt"))
%>
```

- **Clickable**

Thuộc tính quy định cho phép liên kết hay không khi người dùng kích chuột lên hình ảnh.

Cú pháp:

AdRotator.Clickable=false|true

Ví dụ 4.59:

```
<%
set adrot=Server.CreateObject("MSWC.AdRotator")
adrot.Clickable=false
Response.Write(adrot.GetAdvertisement("ads.txt"))
%>
```

- **TargetFrame**

Tên của cửa sổ Frame sẽ hiển thị ảnh quảng cáo này – advertisement

Cú pháp:

AdRotator. TargetFrame = „Cửa sổ Frame“

Ví dụ 4.60:

```
<%
    set adrot=Server.CreateObject("MSWC.AdRotator")
    adrot.TargetFrame="target='_blank'"
    Response.Write(adrot.GetAdvertisement("ads.txt"))
%>
```

4.7.3. Các phương thức của đối tượng AdRotator

Đối tượng AdRotator có một phương thức GetAdvertisement dùng để hiển thị lên trên trang HTML nội dung quảng cáo được chỉ dẫn bởi file text.

Cú pháp:

AdRotator.GetAdvertisement(Filename)

Trong đó Filename là một file dạng text đã được tạo trước đó

Ví dụ 4.61:

```
<%
    set adrot=Server.CreateObject("MSWC.AdRotator")
    Response.Write(adrot.GetAdvertisement("ads.txt"))
%>
```

Ví vụ này sẽ đọc nội dung quảng cáo trong file ads.txt lên trên trang.

4.8. Kết nối cơ sở dữ liệu

ADO là kỹ thuật mới do Microsoft phát triển để làm việc với cơ sở dữ liệu (CSDL), được dùng để cung cấp các khả năng kết nối và xử lý trên CSDL.

Việc xử dụng ADO để truy xuất và xử lý CSDL trong trang ASP có thể chia làm các bước chính sau:

- Kết nối với CSDL thông qua OLEDB hoặc ODBC
- Xây dựng truy vấn dữ liệu và yêu cầu thực hiện câu truy vấn để thực hiện thao tác xử lý trên CSDL
- Xử lý các kết quả trả về từ câu truy vấn

Ngắt kết nối với CSDL, giải phóng các tài nguyên của hệ thống đã dùng

4.8.1. Kết nối với cơ sở dữ liệu

1) Tạo các connection string thông qua OLEDB, ODBC

Connection String là một chuỗi kí tự được dùng để lưu trữ thông tin về dữ liệu như sau:

- Thông tin về hệ quản trị CSDL
- Thông tin về vị trí đặt CSDL
- Mô hình kết nối CSDL: ADO cho phép thông qua OLEDB hoặc ODBC

Bảng liệt kê kết nối OLEDB và ODBC

Data source	**OLEDB**
Microsoft Access	Provider=Microsoft.Jet.OLEDB4.0;Data source = path đến .mdb
Microsoft SQL	Provider=SQLOLEDB.1;Data source = path đến CSDL trên máy chủ
Data source	**ODBC**
Microsoft Access	Driver={Microsoft Access Driver (*.mdb)};DBQ = đường dẫn đến tệp .mdb
Microsoft SQL	Driver={SQL Server.1 };Server= đường dẫn đến CSDL trên server

2) Tạo các connection string thông qua DSN

Có thể xây dựng connection string bằng cách tạo Data source Name (DSN) trong ODBC. Một DSN chứa các thông tin sau:

- Tên của DSN
- Tệp tin CSDL mà nó trỏ tới
- Con trỏ chỉ đến các driver kết nốivới tệp tin CSDL
- UserID và password để truy xuất data store

• Các thông tin cần thiết khác cho kết nối

Trước khi tạo các Script truy xuất cơ sở dữ liệu(CSDL), chúng ta cần chỉ dẫn cho ADO để xác định nguồn dữ liệu cần truy xuất và cách thức liên kết CSDL. Phổ biến và đơn giản nhất đó là sử dụng tên nguồn dữ liệu(DSN) để định vị và cấu hình nguồn dữ liệu tương thích chuẩn ODBC. Với ODBC chúng ta có thể lựa chọn các kiểu DNS để tạo, đó là: User, System hoặc File. Các DNS User và System thường trú trong registry của hệ điều hành WindowsNT. System DNS cho phép tất cả người sử dụng truy nhập vào Server đó đều có thể truy xuất một CSDL, trong khi đó User DNS hạn chế đối với mỗi người sử dụng đăng nhập vào Server; File DSN sẽ lưu thông tin dưới dạng file cho phép nhiều người sử dụng truy xuất CSDL và dễ dàng chuyển từ Server này sang Server khác chỉ bằng việc copy các tệp DSN.

Chúng ta có thể tạo a DSN bằng cách:

- Vào trong Start\Control Panel, click chuột vào biểu tượng ODBC, chọn một dạng DSN

- Click Add, chọn một trình điều khiển dữ liệu(.MDB, SQL)

- Theo các chỉ dẫn trên màn hình để cấu hình DSN cho CSDL của chúng ta

4.8.2. Các đối tượng của ADO

ADO có các đối tượng Connection, Conmmand, Recordset, Record, Stream và tập hợp Errors Fields, Properties, Parameters.

• Đối tượng Connection: cho phép kết nối với CSDL. Nó chứa 3 thông tin :
  - Cơ sở dữ liệu
  - Giao thức (driver/provider) để trao đổi thông tin
  - Username và password

• Đối tượng Command: thực hiện các cậu lệnh SQL

• Đối tượng Recordset: Chứa tập hợp các dữ liệu được rút gọn từ CSDL. Cho phép thay đổi dữ liệu như thêm, xóa, sửa dữ liệu hay di chuyển bản ghi.

• Đối tượng Record: lưu trữ một hàng (bản ghi) trong Recordset, một thư mục hay tập tin trong File system

• Đối tượng Stream: quản lý dữ liệu dạng nhĩ phân, nó được dùng quản lý dữ liệu BLOB (Binary Large Object) như hình ảnh hay mảng dữ liệu lớn

1) Đối tượng Connection

Kết nối CSDL qua đối tượng Connection

Để thiết lập kết nối CSDL, cần thực hiện các bước sau:

• Tạo một thực thể và đối tượng Connection từ phía server bằng câu lệnh : Server.CreateObject(“ADODB.Connection”)

• Sử dụng phương thức Open để mở kết nối CSDL. Tham số của phương thức này lấy từ chuỗi Connection String, chuỗi này tương ứng với CSDL.

Khi cần kết nối cố định cho tất cả các trang ta có thể thiết lập tầm vực cho đối tượng Connection bằng cách viết các thủ tục sau trong tệp tin global.asa

- Ở mức Application:
- Ở mức Session:

Thao tác dữ liệu thông qua đối tượng Connection

Đối tượng Connection cung cấp phương thức Execute để thực hiện câu lệnh truy vấn.

Cú pháp:

objConn.Excute CommandText, RecordAffected, Options

CommandText: câu lệnh SQL, tên bảng hay Stored Procedure

Options: quy định loại CommandText

Hằng	Giá trị	Loại của CommnadText
adCmdUnknown	0	Mặc định, không xác định loại CommnadText
adCmdText	1	CommandText là câu lệnh SQL
adCmdTable	2	CommandText là tên bảng
adCmdStore	4	CommandText là stored procedure hay câu truy vấn

Ví dụ 4.62: Để kết nối tới cơ sở dữ liệu quản lý sinh viên trong hệ quản trị cơ sở dữ liệu Access ta có thể thực hiện như sau:

```
<%
  'khai báo biến kết nối cơ sở dữ liệu
  Dim cnn
  ' tạo đối tượng kết nối
    set cnn = Server.CreateObject("ADODB.Connection")
    ' Tạo xâu kết nối
    cnn.Connectionstring = "Provider=Microsoft.Jet.OLEDB.4.0; Data
Source=D:\asp\quanlysinhvien.mdb"
  ' Mở kết nối
    cnn.open
%>
```

2) Đối tượng Command

Để thao tác và thay đổi cơ sở dữ liệu bằng Command, thực hiệnc các bước sau:

- Khai báo và khởi tạo thực thể (instance) của đối tượng Command
- Khởi tạo các thuộc tính đối tượng

ActiveConnection	Chứa đối tượng Connection đã được khai báo
CommandText	Chứa câu lệnh SQL hay tên bảng
CommandType	Chứa các thuộc tính qui định cho CommandText
CommandTimeout	THời gian thực hiện câu lệnh, nếu việc thực hiện vượt thời gian đã định, nó sẽ thông báo lỗi
Prepard	True/False: True cho phép biên dịch trước khi thực hiện câu lệnh, falsse ngược lại
Execute	Thực thi câu lệnh

Giá trị của thuộc tính CommandType:

Thuộc tính	Mô tả
adCmdType	Câu lệnh SQL
adCMDTable	Tên bảng
adCmdStoreProc	Stored Procuedure hay câu truy vấn
adCmdUnknown	Giá trị mặc định

3) Đối tượng Recordset

Các phương thức của đối tượng Recordset

Phương thức	Diễn giải
Addnew	Tạo mới Record
Cancel	Hủy thao tác đang thực thi
Close	Đóng Recordset và đối tượng liên quan
Delete	Xóa record hay tập record hiện thời
Find	Tìm một record thỏa mãn điều kiện
MoveFirset	Đưa vị trị của record hiện hành về record đầu tiên
MoveLast	Đưa vị trị của record hiện hành về record cuối cùng
MoveNext	Đưa vị trị của record hiện hành về record tiếp theo
MovePrivious	Đưa vị trị của record hiện hành về record trước đó

Open	Mở một recordset
Requery	Cập nhật lại dữ liệu bằng cách thực hiện lại cậu lệnh truy vấn ban đầu
Resync	Làm tuơi lại dữ liệut rong đối tượng Recorset hiện thời
Save	Lưu Recordset vào tệp
Seek	Tìm chỉ mục Recordset
Update	Lưu thay đổi
GetRows	Lấy nhiều record đưa vào một mảng
GetString	Trả về recordset dưới dạng chuỗi

a. Phương thức Open

Cú pháp: **objRs.Open Source, Connection, CursorType, Locktype, Options**

b. Phương thức Addnew

Cho phép tạo một bản ghi mới, gán dữ liệu mới cho các field của các bản ghi và nó chỉ được cập nhật vào CSDL khi ta gọi phương thức Update hay Updatebatch

c. Phương thức Update

Phương thức này dùng để cập nhật bản ghi hiện thời trong CSDL

d. Phương thức Delete

phương thức này cho phép xóa bản ghi trong Recordset

Cú pháp: **objRs.Delete <tham số>**

e. Phương thức Close: Ngắt kết nối với CSDL

Lưu trữ dữ liệu trả về

ADO sử dụng đối tượng Recordset để lưu trữ kết quả trả về từ câu truy vấn dữ liệu SELECT. Vì kết quả trả về của một câu truy vấn SELECT có nhiều bản ghi, cho nên có thể xem Recordset như mảng các bản ghi.

Có 2 cách lấy dữ liệu từ câu truy vấn đặt vào biến Recordset:

Thực hiện phương thức Exexute của đối tượng Connection như câu lệnh truy vấn và trả về kết quả cho Recordset.

Ví dụ: Set rs = Conn.Execute(strSQL)

Tạo một thực thể cho đối tượng Recordset và sử dụng phương thức Open, kết hợp với đối tượng Connection đã tạo.

Hiển thị dữ liệu trả về

Khi muốn lấy dữ liệu của một trường trong bản ghi hiện hành, lấy chuỗi tên của trường đó như là đối số cho đối tượng Recordset hay đối số của thuộc tính Fields của đối tượng Recordset. Ví dụ objRS(“HOTEN”) hay objRS.Fields(“HOTEN”).

Khi muốn dịch chuyển qua lại đến các bản ghi được lưu trong đối tượng Recordset, sử dụng phương thức MoveNext, MovePrevious, MoveFirst, MoveLast và phải đi kèm với việc kiểm tra bản ghi hiện thời đang ở vị trí đầu hay cuối bản ghi. Để làm điều đó, dùng thuộc tính BOF hoặc EOF để kiểm tra.

Ví vụ 4.63: Đọc dữ liệu từ bảng cơ cở dữ liệu sinhvien trong cơ sở dữ liệu Quanlysinhvien.mdb

```
<%
  'khai báo biến kết nối cơ sở dữ liệu
  Dim cnn
  ' tạo đối tượng kết nối
    set cnn = Server.CreateObject("ADODB.Connection")
    ' Tạo xâu kết nối
cnn.Connectionstring = "Provider=Microsoft.Jet.OLEDB.4.0; Data
Source=D:\asp\quanlysinhvien.mdb"
 ' Mở kết nối
    cnn.open
Dim rs, sql
' tạo đối tượng RecordSet
set rs = Server.CreateObject("ADODB.RecordSet")
' Tạo câu lệnh sql
sql = "Select * from sinhvien"
' Mở RecordSet
    rs.open sql, cn, 1, 3
    while (Not rs.eof )
          response.write(rs("masv") &" " & rs("Tensv") &" " & rs("malop")& "<br>"
          )
        rs.MoveNext
    wend
%>
```

4) Đối tượng Record

Đối tượng này lưu trữ bản ghi trong Recordset, một thực mục hay tệp tin trong File System. Để sử dụng đối tượng này cần phải khai báo thực thể cho đối tượng Recordset.

Dim objRec

Set objRec = Server.CreateObject ("ADODB.Record")

Đối tượng Recordset có các phương thức sau:

Phương thức	Diễn giải
Cancel	Hủy thực hiện trên Record
Close	Đóng đối tượng Record
CopyRecord	Copy tệp hay thư mục
DeleteRecord	Xóa một tệp hoặc thư mục
GetChildren	Trả về một đối tượng Recordset, mỗi dòng của Recordset lưu trong tệp hoặc thư mục
MoveRecord	Di chuyển một tệp hay một thư mục
Open	Mở một đối tượng Record tồn tại, tạo mới một tệp hoặc thư mục

a. Phương thức Open

Sau khi tạo thực thể của Record, có thể dùng phương thức này để open, tạo mới một tệp hay tạo mới một thư mục:

Cú pháp*:*

objRec.Open
Source,ActiveConnection,[Model],[CreateOption],[Option],[user],[Passwd])

b. Phương thức CopyRecord; dùng để chép một tệp hay một thư mục đến nơi khác.

Cú pháp:

objRec.CopyRecord (Source, Destionation, Username, Password,opt,async)

c. Phương thức DeleteRecord xóa một tệp hoặc thư mục chỉ định

Cú pháp:

objRec.DeleteRecord(Source,Async)

5) Đối tượng Stream

Dùng để lưu trữ luồng dữ liệu dạng text hoặc nhị phân. Trước khi sử dụng cần phải khai báo

Set objStream = Server.CreateObject(“ADODB.Stream”)

Các phương thức Stream:

Close	Đóng đối tượng Stream
Copyto	Chép một số kí tự hoặc byte từ đối tượng Stream này sáng đối tượng Stream khác
Flush	Gửi nội dung của đối tượng Stream
LoadFromFile	Lấy nội dung của một đối tượng vào đối tượng Stream
Open	Mở một đối tượng stream từ URL hay đối tượng Record
Read	Đọc số lượng byte trong đối tượng Stream chứa dữ liệu nhị phân
ReadText	Đọc số kí tự trong đối tượng Stream chứa nội dung là text
SaveToFile	Lưu nội dụng đối tượng Stream vào tệp
SetEOS	Thiết lập thuộc tính của EOS đối với vị strí hiện thời
SkipLine	Bỏ qua một dòng khi đọc một Text Stream
Write	Ghi một số lượng byte của dữ liệu nhị phần và đối tượng Stream
WriteText	Ghi dữ liệu dạng Text vào đối tượng Stream

4.8.3. Tập hợp các Errors

Tập hợp Errors (Errors Collection) chứa tất cả các đối tượng Errors (Error objects). Tất cả các đối tượng Errors thuộc về hệ điều hành riêng kết hợp với ADO. Tập hợp Errors là chức năng chỉ có hiệu lực với các cơ sở dữ liệu đặc biệt như SQL server. Điều này đặc biệt có lợi khi thực hiện các câu lệnh có hơn nhiều một lỗi.

Tập hợp Errors có một phương thức và hai thuộc tính:

- Clear method : Phương thức xoá bỏ tất cả các đối tượng Errors trong tập hợp. Phương thức này được sử dụng để xoá bỏ hoàn toàn các lỗi.

- Count property : Thuộc tính trả lại số của đối tượng Errors hiện tại có trong tập hợp.

- Item property : Thuộc tính trả lại đối tượng Errors được chỉ định.

Đối tượng Error

Chứa chi tiết lỗi ADO bị máy chủ bắt lỗi của nhà cung cấp dữ liệu. Tập này được xoá bỏ mỗi lần hoạt động mới gây ra lỗi. Từ đối tượng Errors có thể truy tìm tên, số và miêu tả của lỗi gây ra.

Các thuộc tính của đối tượng Error

Description: chuỗi miêu tả gắn với lỗi.

HelpContext: giá trị ngữ cảnh trong file Help (nếu thuộc tính HelpFile kèm theo chỉ rằng có file trợ giúp Windows kết hợp với đối tượng Errors).

HelpFile: chuỗi đường dẫn và tên file của file Help Windows.

NativeError: mã lỗi do nhà cung cấp dữ liệu địa phương gây ra.

Number: số lỗi đối với đối tượng Error (bằng 0 nếu không có lỗi).

Source: chuỗi đại diện cho tên đối tượng hoặc ứng dụng gây lỗi ADO.

SQLState: mã lỗi 5 ký tự do nhà cung cấp trả lại.

Ví dụ 4.64: đoạn mã sau đây đọc các và hiển thị các lỗi

```
<%
        For Each dbError In Conn.Errors
                Response.Write "<TR><TD>" & dbError.Description & "</TD>"
                Response.Write "<TD>" & CStr(dbError.NativeError) & "</TD>"
                Response.Write "<TD>" & dbError.SQLState & "</TD></TR>"
        Next
        Conn.Errors.Clear
%>
```

4.8.4. Stored Procedure và truyền tham số

Thủ tục lưu trữ là một đối tượng trong CSDL, bao gồm nhiều câu lệnh T-SQL được tập hợp lại với nhau thành một nhóm, và tất cả các lệnh này sẽ được thực thi khi thủ tục lưu trữ được thực thi.Với thủ tục lưu trữ, một phần nào đó khả năng của ngôn ngữ lập trình được đưa vào trong ngôn ngữ SQL.

Thủ tục lưu trữ có thể có các thành phần sau: Các cấu trúc điều khiển (IF, WHILE, FOR) có thể được sử dụng trong thủ tục. Bên trong thủ tục lưu trữ có thể sử dụng các biến như trong ngôn ngữ lập trình nhằm lưu giữ các giá trị tính toán được, các giá trị được truy xuất được từ cơ sở dữ liệu. Một tập các câu lệnh SQL được kết hợp lại với nhau thành một khối lệnh bên trong một thủ tục. Một thủ tục có thể nhận các tham số truyền v ào cũng như có thể trả về các giá trị thông qua các tham số (như trong các ngôn ngữ lập trình). Khi một thủ tục lưu trữ đã được định nghĩa, nó có thể được gọi thông qua tên thủ tục, nhận các tham số truyền vào, thực thi các câu lệnh SQL bên trong thủ tục và có thể trả về các giá trị sau khi thực hiện xong.

1) Lợi ích của việc sử dụng thủ tục lưu trữ:

SQL Server chỉ biên dịch các thủ tục lưu trữ một lần và sử dụng lại kết quả biên dịch này trong các lần tiếp theo trừ khi người dùng có những thiết lập khác. Việc sử dụng lại kết

quả biên dịch không làm ảnh hưởng đến hiệu suất hệ thống khi thủ tục lưu trữ được gọi liên tục nhiều lần.

Thủ tục lưu trữ được phân tích, tối ưu khi tạo ra nên việc thực thi chúng nhanh hơn nhiều so với việc phải thực hiện một tập rời rạc các câu lệnh SQL tương đương theo cách thông thường. Thủ tục lưu trữ cho phép chúng ta thực hiện cùng một yêu cầu bằng một câu lệnh đơn giản thay vì phải sử dụng nhiều dòng lệnh SQL. Điều này sẽ làm giảm thiểu sự lưu thông trên mạng.

Thay vì cấp phát quyền trực tiếp cho ng ười sử dụng trên các câu lệnh SQL và trên các đối tượng cơ sở dữ liệu, ta có thể cấp phát quyền cho người sử dụng thông qua các thủ tục lưu trữ, nhờ đó tăng khả năng bảo mật đối với hệ thống.

Các thủ tục lưu trữ trả về kết quả theo 4 cách:

- Sử dụng các tham số output.
- Sử dụng các lệnh trả về giá trị, các lệnh này luôn trả về giá trị số nguyên.
- Tập các giá trị trả về của mỗi câu lệnh SELECT có trong thủ tục lưu trữ hoặc của quá trình gọi một thủ tục lưu trữ khác trong một thủ tục lưu trữ.
- Một biến con trỏ toàn cục có thể tham chiếu từ bên ngoài thủ tục.

2) Tạo thủ tục lưu trữ

Thủ tục lưu trữ được tạo thông qua câu lệnh Create Procedure.

Cú pháp như sau:

Create Procedure tên_thủ_tục [(danh_sách_tham_số)]

[With Recompile|Encryption|Recompile,Encryption]

As

Các_câu_lệnh_của_thủ_tục

Trong đó:

With Recompile: yêu cầu SQL Server biên dịch lại thủ tục lưu trữ mỗi khi được gọi.

With Encryption: yêu cầu SQL Server mã hóa thủ tục lưu trữ.

Các_câu_lệnh_của_thủ_tục: Các lệnh T-SQL. Các lệnh này có thể nằm trong cặp BEGIN…END hoặc không.

3) Truyền tham số

Cách viết như trên có hạn chế là: trong quá trình làm việc sẽ có rất nhiều đơn đặt hàng mới, do đó người dùng sẽ phải viết đi viết lại những câu lệnh tương tự nhau cho các khách hàng khác nhau. Một cách giải quyết vấn đề này là dùng thủ tục lưu trữ và dùng tham số để nhận các thông tin thay đổi.

Ví dụ 4.65:

create procedure sp_InsertOrderAndOrderDetail

@customerid int,

@orderdate datetime,

@orderid int,

@itemid int,

@quantity decimal,

as

begin

insert into orders

values(@customerid, @orderdate)

insert into orderdetail

values(@orderid, @itemid, @quantity)

end

Thực hiện thủ tực lưu trữ này như sau:

sp_InsertOrderAndOrderDetail '3', '22/7/2008', '4', '1', '10')

4) Thực thi một Store Procedure

Thủ tục lưu trữ được gọi theo cấu trúc

Tên_thủ_tục_lưu _trữ [danh_sách_tham_số]

Cần lưu ý là danh sách tham số truyền vào trong lời gọi phải theo đúng thứ tự khai báo

các tham số trong thủ tục lưu trữ.

Nếu thủ tục được gọi từ một thủ tục khác, thực hiện b ên trong một trigger hay phối hợp

với câu lệnh SELECT, cấu trúc như sau;

Exec Tên_thủ_tục_lưu _trữ [danh_sách_tham_số]

Ví dụ 4.66: Thực thi một procedure *sp_InsertOrderAndOrderDetail* bằng asp

```
<%
Dim customerid,orderdate, orderid, itemid, quantity
 customerid = 4
 orderdate = "22/7/2008"
 orderid = 4
 itemid = 1
 quantity = 10
 sql = "EXEC sp_InsertOrderAndOrderDetail " &_
   " @ customerid =" & customerid & "," &_
   " @ orderdate ='" & orderdate & "'," &_
   " @ orderid ='" & orderid & "'," &_
```

```
  " @ itemid =" & itemid & "," &_
  " @ quantity =" & quantity
 set conn = CreateObject("ADODB.Connection")
 conn.open "<connection string>"
 conn.execute sql, , 129
 Response.write("Record updated.")
 conn.close
 set conn = nothing
%>
```

Câu hỏi và Bài tập chương 4

1. Dùng câu lệnh Response.Write() để tạo ra giao diện như sau

Student List

Studen ID	Student Name	Address
A000838	Nguyen Van A	Ha noi
A00456	Vu Thi B	Hai phong

2. Dùng vòng lặp for để hiển thị các dòng tiêu đề như sau (hiển thị qua thẻ <h>)

Header 1

Header 2

Header 3

Header 4

Header 5

Header 6

3. Viết chương trình bằng asp hiện ra màn hình ngày hiện hành trong tuần bằng tiếng việt.
4. Viết chương trình in ra màn hình 10 dòng chữ Helloworld kích thước tăng dần
5. Tạo CSDL **EShop** trong SQL Server hoặc Access gồm các bảng sau:

-Tạo bảng **catalogs**

	Column Name	Data Type	Length	Allow Nulls
🔑	catalogID	nvarchar	50	
	catalogName	nvarchar	50	✓

-Tạo bảng **products**

	Column Name	Data Type	Length	Allow Nulls
🔑	productID	smallint	2	
	productName	nvarchar	50	✓
	productDesc	ntext	16	✓
	productImg	nvarchar	100	✓
	productPrice	money	8	✓
	productWt	money	8	✓
	numInStock	smallint	2	✓
	catalogID	nvarchar	50	✓
	dateAvailable	smalldatetime	4	✓

-Tạo bảng **customers**

	Column Name	Data Type	Length	Allow Nulls
🔑	customerID	int	4	
	fName	nvarchar	50	✓
	lName	nvarchar	50	✓
	email	nvarchar	50	✓
	address1	nvarchar	50	✓
	address2	nvarchar	50	✓
	city	nvarchar	50	✓
	postalcode	nvarchar	50	✓
	country	nvarchar	50	✓
	phone	nvarchar	50	✓
	status	nvarchar	50	✓
	pass	nvarchar	50	✓

6. Tạo trang kết nối tới CSDL (Connection.asp) cho phép kết nối đến cơ sở dữ liệu đã tạo trong câu 5
7. Tạo trang hiển thị dữ liệu (DisplayCategories.asp) trang này hiển thị tất cửa các loại sản phẩm của cửa hàng, nó cho phép người sử dụng nhấn vào đó để xem các sản phẩm thuộc loại đó.
8. Tạo trang hiển thị sản phẩm theo từng loại (productsByCateogry.asp) trang này hiển thị danh sách sản phẩm theo từng loại, Khi nhấn vào đường link ở trang danh mục loại sản phẩm, thì trang này sẽ hiển thị ra.
9. Tạo trang hiển thị thông tin sản phẩm chi tiết (productDetail.asp) trang này hiển thị thông tin chi tiết của từng sản phẩm khi bạn nhấn vào đường link trang trước.
10. Tạo trang nhập thông tin sản phẩm (ProductNew.asp) cho phép nhập thông tin vào bảng sản phẩm.
11. Tạo trang nhập dữ liệu (InsertCategories.asp) cho phép nhập vào một loại sản phẩm mới cho cửa hang.
12. Tạo trang tìm kiếm thông tin về sản phẩm (FindProduct.asp) cho phép tìm kiếm thông tin về mặt hàng theo tên mặt hang.
13. Tạo trang login (Signin.asp) trang này hiển form login để admin login.
14. Tạo Trang xử lý thông tin login (LoginVerify.asp) trang này xử lý thông tin login, nếu hợp lệ thì sẽ chuyển hướng sang trang dành cho user, nếu không sẽ thông báo lỗi.
15. Tạo trang nhập thông tin đăng ký của khách hàng (RegisterForm.asp) trang này dùng để hiển thị form đăng ký, nhập dữ liệu ở đây sau đó gọi tời file RegisterAction.
16. Tạo Trang xử lý thông tin đăng ký (RegisterAction.asp) trang này dùng để đưa thông tin đăng ký của khách hàng vào trong database.
17. Tạo Trang dành cho khách hàng (UserInfo.asp) trang này dùng để hiển thị thông tin dành cho admin, nếu bạn chưa đăng nhập thì sẽ chuyển hướng về trang đăng nhập.
18. Tạo Trang Logout (Logout.asp) trang này dùng để logout, nó sẽ hủy bỏ session của khách hàng, hủy bỏ thông tin đăng nhập của khách hàng.

TÀI LIỆU THAM KHẢO

[1]. Nguyễn Đức Hải, Nguyễn Phương Lan, Lê Hữu Đạt, "Giáo trình lý thuyết và bài tập ASP", NXB Lao động - Xã hội

[2]. Hoàng Mạnh Hùng, Giáo trình lập trình web, Đại học Đà Lạt

[3]. Phạm Hữu Khang, Hoàng Đức Hải, Phương Lan, "Giáo trình lập trình web bằng ASP 3.0", NXB Lao động - Xã hội

[4]. Bùi Thu Giang, Lê Hoàng Nhân, Nguyễn Trường Sinh, Việt Dũng, Thiết kế trang web với Microsoft FrontPage, NXB Lao động - Xã hội

[5]. Jerry Honeycutt, Special Edition Using HTML 4, Macmillan Computer Publishing

[6]. WallPearl, Simple CSS Standard Edition, WallPearl'sBlog, 2008