

VDOM
TECHNOLOGY



```

import SOAPpy, md5, re, sys
from SOAPpy import SOAP
from xml.dom import Node
from xml.dom.minidom import parseString

saddr = "http://192.168.0.13"
appid = "f8e32f41-320e-4e71-8910-b3d2476b92ee"

# session protector class
class VDOM_session_protector:
    """class used to protect web services from unauthorized access"""

    def __init__(self, hash_str):
        """constructor"""
        self.__hash = hash_str

    def next_session_key(self, session_key):
        """generate next session key"""
        ## verify hashcode
        if self.__hash == "":
            raise VDOM_exception("hash code is empty")

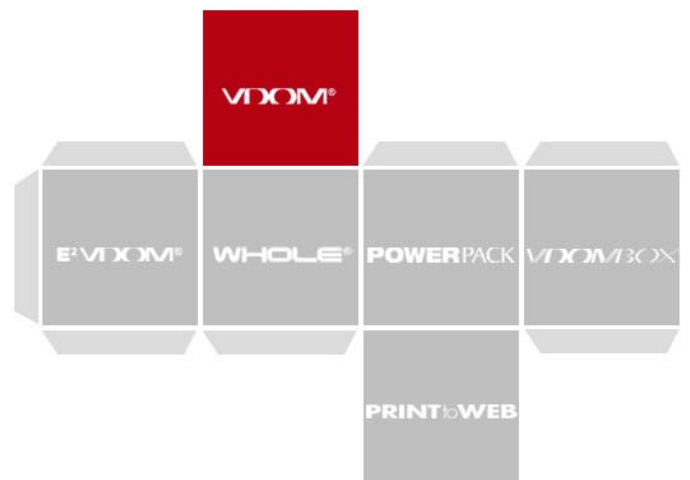
        for idx in xrange(len(self.__hash)):
            i = self.__hash[idx]
            if not str(i).isdigit():
                raise VDOM_exception("hash code contains non-digit letter '%s'" % str(i))

        result = 0
        for idx in xrange(len(self.__hash)):
            i = self.__hash[idx]
            result += int(self.__calc_hash(session_key, int(i)))

        return ["0"*10 + str(result)[0:10]

    def __calc_hash(self, session_key, int(i)):
        """calculate hash code"""
        return int(md5("%s%s" % session_key + str(i)).hexdigest(), 16)
    
```

Vscript Quick Start



VDOM®

E-VDOM®

WHOLE®

POWERPACK

VDOM/BOX

PRINTtoWEB

Document version:	Beta 0.9
Edition Date :	08/10/08
Langage :	EN

Copyright © 2008 VDOM Box International OOD, All rights reserved.

The information published in this document represents the opinion of VDOM Box International as for the subjects tackled at the date of publication. VDOM Box International having to answer fluctuating conditions of market, the information published in this document should not be interpreted like an engagement of its share, VDOM Box International which cannot guarantee the precision of this information after the date of publication.

This guide of evaluation is published only at ends of information. VDOM Box International does not offer any guarantee, express or implicit, as for the contents of this document.

The subject tackled in this document can make the object of patents, patent applications, marks, copyright or other rights of ownership intellectual belonging to VDOM International Box and/or Nicolas Korboulewsky. This document does not give you any right to these patents, marks, copyright or another intellectual property, except in the case of a specific contract drawn up with VDOM International Box and/or Nicolas Korboulewsky.

V.D.O.M., logo V.D.O.M., software suite, V.D.O.M., W.H.O.L.E. logo W.H.O.L.E. E²V.D.O.M., Logo E²V.D.O.M. are marks of trade or trademarks under license by VDOM International Box in France and/or in other countries. The other names or marks belong to their respective owners.

VDOM Box International, Sofia 1220, Nadezkda bl 122 Ent. A

Table of Content

1. Introduction

1.1. What is Vscript ?.....	6
1.2. Easy to use	6

2. Basis of the language

2.1. VScript Data Type	7
2.2 Variant Subtype	7
2.3.Variables	8
2.3.1. Declaring variables	8
2.3.2. Scope and Lifetime of variables	8
2.3.3. Assigning values to variables	8
2.3.4. Scalar variables and array variables	8
2.4. Constants	10
2.4.1. Creating Constants	10
2.5. Operators.	10
2.5.1. Operator precedence	11
2.6. Using Conditional Statements	11
2.6.1. Controlling Program Execution	11
2.6.2. Making decisions using If ... Then ... Else	11
2.6.3. Running statements if a condition is True	12
2.6.4. Running statements if a condition is true and other if it's false	12
2.6.6. Deciding between several alternatives	12
2.6.7. Make decision with select case	13
2.7. Looping Through Code	14
2.7.1. Using Do ... Loop	14
2.7.2. Repeating a statement while a condition is true	14
2.7.3. Repeating a statement until a condition becomes true	15
2.7.4. Exiting a Do Loop Statement from inside the Loop	16
2.7.5. Using While Wend	16
2.7.6. Using For Next	16
2.7.7. Using For Each Next	18
2.8. VScript Procedures	19
2.8.1. Sub procedures	19

2.8.2. Function procedures	19
2.8.3. Getting data into and out of procedures	19
2.8.4. Using Sub and Function Procedures in Code	20
2.9 VScript Classes	21
2.9.1. Define classes and create instances	21
2.9.2. Using Class Procedures	21
2.9.3. Defining and using Properties	22
2.9.4. Default property	22

3. VScript & VDOM Objects / Server Objects

3.1. VDOM Objects	24
3.1.1. Creating objects	24
3.1.2. Using objects	25
3.1.3. Get acces to foreign objects	25
3.2. Embedded Objects	26
3.2.1. Server	26
3.2.2. RegExp	26
3.3. VScript HTTP Obejcts	28
3.3.1 Request	28
3.3.2 Response	29
3.3.3 Session	29

4. VScript & VDOM DB Acces

4.1. General	31
4.2. Connection	31
4.2.1 Open a connection	31
4.2.2 Close a connection	31
4.3. Query the DB	32
4.3.1 Query that return data	32
4.3.1 Query that return no data	32

5. Cookbook

5.1. Page access control	34
--------------------------------	----

Introduction

What Is VScript?

VDOM Server provides perfect possibilities of fast and easy creation of web applications. But at one moment you will find yourself in need of additional functionalities, not supported by standard components. In this case you may use the possibility to write scripts with extended functionality.

VScript is the script language of VDOM Box Server. It's similar to Microsoft's VBScript syntax and implements most of functions, types and objects of this language. We provide VScript with partial support of ASP technology.

Easy to Use

Our solution allows easy migration for VBScript/ASP developers to our platform. They can start developing web applications without significant time losses for adaptation to new language and environment.

Simple syntax, useful set of functions and good usability are additional advantages for novice developers.

Basis of the language

VScript Data Types

As the VBScript, VScript has only one data type called a Variant. A Variant is a special kind of data type that can contain different kinds of information, depending on how it is used. Because Variant is the only data type in VScript, it is also the data type returned by all functions in VScript.

At its simplest, a Variant can contain either numeric or string information. A Variant behaves as a number when you use it in a numeric context and as a string when you use it in a string context. That is, if you are working with data that looks like numbers, VScript assumes that it is numbers and does what is most appropriate for numbers. Similarly, if you're working with data that can only be string data, VScript treats it as string data. You can always make numbers behave as strings by enclosing them in quotation marks (" ").

Variant Subtypes

Beyond the simple numeric or string classifications, a Variant can make further distinctions about the specific nature of numeric information. For example, you can have numeric information that represents a date or a time. When used with other date or time data, the result is always expressed (with a little exceptions) as a date or a time. You can also have a rich variety of numeric information ranging in size from Boolean values to huge floating-point numbers. These different categories of information that can be contained in a Variant are called subtypes. Most of the time, you can just put the kind of data you want in a Variant, and the Variant behaves in a way that is most appropriate for the data it contains.

VScript support following subtypes:

Type	Description
Empty	Default value for uninitialized variant. Value is 0 for numeric variables or a zero-length string ("") for string variables.
Null	Contains no valid data. This value is intentionally set like this.
Boolean	Contains either True (equivalent to 0) or False (equivalent to -1).
Integer	Contains integer in the range -2,147,483,648 to 2,147,483,647.
Double	Contains a double-precision, floating-point number in the range -1.79769313486232E308 to -4.94065645841247E-324 for negative values; 4.94065645841247E-324 to 1.79769313486232E308 for positive values.
Date (Time)	Contains a number that represents a date between January 1, 100 to December 31, 9999.
String	Contains a variable-length string that can be up to approximately 2 billion characters in length.
Object	Contains an object.

Table 1: VScript data types

Variables

A variable is a convenient placeholder that refers to a computer memory location where you can store program information that may change during the time your script is running. In VScript, variables are always of one fundamental data type, Variant.

Declaring Variables

You declare variables explicitly in your script using the Dim statement. For example:

```
| Dim DegreesFahrenheit
```

You declare multiple variables by separating each variable name with a comma. For example:

```
| Dim Top, Bottom, Left, Right
```

Scope and Lifetime of Variables

A variable's scope is determined by where you declare it. When you declare a variable within a procedure, only code within that procedure can access or change the value of that variable. It has local scope and is a procedure-level variable. If you declare a variable outside a procedure, you make it recognizable to all the procedures in your script. This is a script-level variable, and it has script-level scope.

The lifetime of a variable depends on how long it exists. The lifetime of a script-level variable extends from the time it is declared until the time the script is finished running. At procedure level, a variable exists only as long as you are in the procedure. When the procedure exits, the variable is destroyed. Local variables are ideal as temporary storage space when a procedure is executing. You can have local variables of the same name in several different procedures because each is recognized only by the procedure in which it is declared.

Assigning Values to Variables

Values are assigned to variables creating an expression as follows: the variable is on the left side of the expression and the value you want to assign to the variable is on the right as in the many other languages. For example:

```
| B = 200
```

Scalar Variables and Array Variables

A variable containing a single value is a scalar variable. Other possible case is an array variable that can contain many indexed values. Array variables and scalar variables are declared in the same way, except that the declaration of an array variable uses parentheses () following the variable name. In the following example, a single-dimension array containing 11 elements is declared:


```
| Dim A(9)
```

Although the number shown in the parentheses is 9, all arrays in VScript are zero-based, so this array actually contains 10 elements. In a zero-based array, the number of array elements is always the number shown in parentheses plus one. This kind of array is called a fixed-size array.

You assign data to each of the elements of the array using an index into the array. Beginning at zero and ending at 9, data can be assigned to the elements of an array as follows:

```
| A(0) = 256  
| A(1) = 324  
| A(2) = 100  
| ...  
| A(9) = 55
```

Similarly, the data can be retrieved from any element using an index into the particular array element you want. For example:

```
| ...  
| SomeVariable = A(8)  
| ...
```

Arrays aren't limited to a single dimension. You can have as many as 60 dimensions, although most people can't comprehend more than three or four dimensions. You can declare multiple dimensions by separating an array's size numbers in the parentheses with commas. In the following example, the MyTable variable is a two-dimensional array consisting of 6 rows and 11 columns:

```
| ...  
| Dim MyTable (5, 10)  
| ...
```

In a two-dimensional array, the first number is always the number of rows; the second number is the number of columns.

You can also declare an array whose size changes during the time your script is running. This is called a dynamic array. The array is initially declared within a procedure using either the Dim statement or using the ReDim statement. However, for a dynamic array, no size or number of dimensions is placed inside the parentheses. For example:

```
| Dim MyArray()  
| Redim AnotherArray()
```

To use a dynamic array, you must subsequently use `ReDim` to determine the number of dimensions and the size of each dimension. In the following example, `ReDim` sets the initial size of the dynamic array to 25. A subsequent `ReDim` statement resizes the array to 30, but uses the `Preserve` keyword to preserve the contents of the array as the resizing takes place.

```
ReDim MyArray(25)
...
ReDim Preserve MyArray(30)
...
```

There is no limit to the number of times you can resize a dynamic array, although if you make an array smaller, you lose the data in the eliminated elements.

Constants

A constant is a meaningful name that takes the place of a number or string and never changes.

Creating Constants

You create user-defined constants in VScript using the `Const` statement. Using the `Const` statement, you can create string or numeric constants with meaningful names and assign them literal values. For example:

```
Const MyString = "This is my string."
Const MyAge = 49
...
```

Note that the string literal is enclosed in quotation marks (" "). Quotation marks are the most obvious way to differentiate string values from numeric values. You represent Date literals and time literals by enclosing them in number signs (#). For example:

```
Const CutOffDate = #6-1-97#
```

You may want to adopt a naming scheme to differentiate constants from variables. This will prevent you from trying to reassign constant values while your script is running. For example, you might want to use a "vb" or "con" prefix on your constant names, or you might name your constants in all capital letters. Differentiating constants from variables eliminates confusion as you develop more complex scripts.

Operators.

VScript has a full range of operators, including arithmetic operators, comparison operators, concatenation operators, and logical operators as a VBScript.

Operator Precedence

When several operations occur in an expression, each part is evaluated and resolved in a predetermined order called operator precedence. You can use parentheses to override the order of precedence and force some parts of an expression to be evaluated before others. Operations within parentheses are

always performed before those outside. Within parentheses, however, standard operator precedence is maintained.

When expressions contain operators from more than one category, arithmetic operators are evaluated first, comparison operators are evaluated next, and logical operators are evaluated last. Comparison operators all have equal precedence; that is, they are evaluated in the left-to-right order in which they appear. Arithmetic and logical operators are evaluated in the following order of precedence.

Arithmetic: Exponentiation (^), Unary negation (-), Multiplication (*), Division (/), Integer division (\), Modulus arithmetic (Mod), Addition (+), Subtraction (-), String concatenation (&).

Logical: Logical negation (Not), Logical conjunction (And), Logical disjunction (Or), Logical exclusion (Xor), Logical equivalence (Eqv), Logical implication (Imp).

When multiplication and division occur together in an expression, each operation is evaluated as it occurs from left to right. Likewise, when addition and subtraction occur together in an expression, each operation is evaluated in order of appearance from left to right.

The string concatenation (&) operator is not an arithmetic operator, but in precedence it falls after all arithmetic operators and before all comparison operators. The Is operator is an object reference comparison operator. It does not compare objects or their values; it checks only to determine if two object references refer to the same object.

Using Conditional Statements

Controlling Program Execution

You can control the flow of your script with conditional statements and looping statements. Using conditional statements, you can write VScript code that makes decisions and repeats actions. The following conditional statements are available in VScript:

- If...Then...Else statement
- Select Case statement

Making Decisions Using If...Then...Else

The If...Then...Else statement is used to evaluate whether a condition is True or False and, depending on the result, to specify one or more statements to run. Usually the condition is an expression that uses a comparison operator to compare one value or variable with another. For information about comparison operators, see Comparison Operators. If...Then...Else statements can be nested to as many levels as you need.

Running Statements if a Condition is True

To run only one statement when a condition is True, use the single-line syntax for the If...Then...Else statement. The following example shows the single-line syntax. Notice that this example omits the Else keyword.

```
Sub FixDate()  
    Dim myDate  
    myDate = #2/13/95#  
    If myDate < Now then myDate = Now  
End Sub
```

To run more than one line of code, you must use the multiple-line (or block) syntax. This syntax includes the End If statement, as shown in the following example:

```
Sub AlertUser(value)  
    If value = 0 Then  
        this.Label.color = "FF0000"  
        this.Label.Font = "bold 12pt"  
    End if  
End Sub
```

Running Certain Statements if a Condition is True and Running Others if a Condition is False

You can use an If...Then...Else statement to define two blocks of executable statements: one block to run if the condition is True, the other block to run if the condition is False.

```
Sub AlertUser(value)  
    If value = 0 Then  
        this.Label.color = "FF0000"  
        this.Label.Font = "bold 12pt"  
    Else  
        this.Label.color = "FFFF00"  
        this.Label.Font = "bold 10pt"  
    End if  
End Sub
```

Deciding Between Several Alternatives

A variation on the If...Then...Else statement allows you to choose from several alternatives. Adding ElseIf clauses expands the functionality of the If...Then...Else statement so you can control program flow based on different possibilities.

For example:

```
Sub AnalyseCardType  
    CardType=Request.Form("CardType")  
    If CardType = "MasterCard" Then  
        DisplayMCLogo  
        ValidateMCAccount  
    ElseIf CardType = "Visa" Then  
        DisplayVisaLogo  
        ValidateVisaAccount
```

```
ElseIf CardType = "American Express" Then
    DisplayAMEXCOLogo
    ValidateAMEXCOAccount
Else
    DisplayUnknownLogo
    PromptAgain
End if
End Sub
```

You can add as many Elseif clauses as you need to provide alternative choices. Extensive use of the Elseif in clauses often becomes cumbersome. A better way to choose between several alternatives is the Select Case statement.

Making Decisions with Select Case

The Select Case structure provides an alternative to If...Then...Elseif for selectively executing one block of statements from among multiple blocks of statements. A Select Case statement provides capability similar to the If...Then...Else statement, but it makes code more efficient and readable.

A Select Case structure works with a single test expression that is evaluated once, at the top of the structure. The result of the expression is then compared with the values for each Case in the structure. If there is a match, the block of statements associated with that Case is executed, as in the following example.

```
Select Case Request.Form("CardType")
    Case "MasterCard"
        DisplayMCLogo
        ValidateMCAccount
    Case "Visa"
        DisplayVisaLogo
        ValidateVisaAccount
    Case "American Express"
        DisplayAMEXCOLogo
        ValidateAMEXCOAccount
    Case Else
        DisplayUnknownLogo
        PromptAgain
End Select
```

Notice that the Select Case structure evaluates an expression once at the top of the structure. In contrast, the If...Then...Elseif structure can evaluate a different expression for each Elseif statement. You can replace an If...Then...Elseif structure with a Select Case structure only if each Elseif statement evaluates the same expression.

Looping Through Code

Looping allows you to run a group of statements repeatedly. Some loops repeat statements until a condition is False; others repeat statements until a condition is True. There are also loops that repeat statements a specific number of times.

The following looping statements are available in VBScript:

- Do...Loop: Loops while or until a condition is True.
- While...Wend: Loops while a condition is True.
- For...Next: Uses a counter to run statements a specified number of times.
- For Each...Next: Repeats a group of statements for each item in a collection or each element of an array.

Using Do Loops

You can use Do...Loop statements to run a block of statements an indefinite number of times. The statements are repeated either while a condition is True or until a condition becomes True.

Repeating Statements While a Condition is True

Use the While keyword to check a condition in a Do...Loop statement. You can check the condition before you enter the loop (as shown in the following ChkFirstWhile example), or you can check it after the loop has run at least once (as shown in the ChkLastWhile example). In the ChkFirstWhile procedure, if myNum is set to 9 instead of 20, the statements inside the loop will never run. In the ChkLastWhile procedure, the statements inside the loop run only once because the condition is already False.

```
Sub ChkFirstWhile()  
    Dim counter, myNum  
    Counter = 0  
    myNum = 20  
    Do While myNum > 10  
        myNum = myNum - 1  
        counter = counter + 1  
    Loop  
    this.Label.value =_  
        "The loop made " & counter & " repetitions."  
End Sub
```

```
Sub ChkLastWhile()  
    Dim counter, myNum  
    Counter = 0  
    myNum = 9  
    Do  
        myNum = myNum - 1  
        counter = counter + 1
```

```

        Loop While myNum > 10
        this.Label.value =_
            "The loop made " & counter & " repetitions."
    End Sub

```

Repeating a Statement Until a Condition Becomes True

There are two ways to use the Until keyword to check a condition in a Do...Loop statement. You can check the condition before you enter the loop (as shown in the following ChkFirstUntil example), or you can check it after the loop has run at least once (as shown in the ChkLastUntil example). As long as the condition is False, the looping occurs.

```

Sub ChkFirstUntil()
    Dim counter, myNum
    Counter = 0
    myNum = 20
    Do Until myNum = 10
        myNum = myNum + 1
        counter = counter + 1
    Loop
    this.Label.value =_
        "The loop made " & counter & " repetitions."
End Sub

```

```

Sub ChkLastUntil()
    Dim counter, myNum
    Counter = 0
    myNum = 1
    Do
        myNum = myNum + 1
        counter = counter + 1
    Loop While myNum = 10
    this.Label.value =_
        "The loop made " & counter & " repetitions."
End Sub

```

Exiting a Do...Loop Statement from Inside the Loop

You can exit a Do...Loop by using the Exit Do statement. Because you usually want to exit only in certain situations, such as to avoid an endless loop, you should use the Exit Do statement in the True statement block of an If...Then...Else statement. If the condition is False, the loop runs as usual.

In the following example, myNum is assigned a value that creates an endless loop. The If...Then...Else statement checks for this condition, preventing the endless repetition.

```

Sub ExitExample()
    Dim counter, myNum
    Counter = 0
    myNum = 9
    Do Until myNum = 0
        myNum = myNum - 1
        counter = counter + 1
        If myNum < 5 Then Exit Do
    Loop
    this.Label.value = _
        "The loop made " & counter & " repetitions."
End Sub

```

Using While...Wend

The While...Wend statement is provided in VBScript for those who are familiar with its usage. However, because of the lack of flexibility in While...Wend, it is recommended that you use Do...Loop instead.

Using For...Next

You can use For...Next statements to run a block of statements a specific number of times. For loops, use a counter variable whose value increases or decreases with each repetition of the loop.

The following example causes a procedure called MyProc to execute 50 times. The For statement specifies the counter variable x and its start and end values. The Next statement increments the counter variable by 1.

```

Sub DoMyProc50Times()
    Dim x
    For x = 1 To 50
        myProc
    Next
End Sub

```

Using the Step keyword, you can increase or decrease the counter variable by the value you specify. In the following example, the counter variable j is incremented by 2 each time the loop repeats. When the loop is finished, the total is the sum of 2, 4, 6, 8, and 10.

```

Sub TwosTotal()
    Dim j, total
    For j = 2 To 10 Step 2
        total = total + j
    Next
    this.Label.value = _
        "The total is " & total
End Sub

```


To decrease the counter variable, use a negative Step value. You must specify an end value that is less than the start value. In the following example, the counter variable myNum is decreased by 2 each time the loop repeats. When the loop is finished, total is the sum of 16, 14, 12, 10, 8, 6, 4, and 2.

```
Sub NewTotal()  
    Dim myNum, total  
    For myNum = 16 To 2 Step -2  
        total = total + myNum  
    Next  
    this.Label.value =_  
        "The total is " & total  
End Sub
```

You can exit any For...Next statement before the counter reaches its end value by using the Exit For statement. Because you usually want to exit only in certain situations, such as when an error occurs, you should use the Exit For statement in the True statement block of an If...Then...Else statement. If the condition is False, the loop runs as usual.

Using For Each...Next

A For Each...Next loop is similar to a For...Next loop. Instead of repeating the statements a specified number of times, a For Each...Next loop repeats a group of statements for each item in a collection of objects or for each element of an array. This is especially helpful if you don't know how many elements are in a collection.

```
For each name in request.servervariables  
    result=result & name & "-" &_  
        Request.servervariables(name)  
Next
```

VScript Procedures

In VScript, there are two kinds of procedures; the Sub procedure and the Function procedure.

Sub Procedures

A Sub procedure is a series of VScript statements (enclosed by Sub and End Sub statements) that perform actions but don't return a value. A Sub procedure can take arguments (constants, variables, or expressions that are passed by a calling procedure). If a Sub procedure has no arguments, its Sub statement must include an empty set of parentheses ().

```
Sub ConvertTemp()  
    temp = request.QueryString("Degrees")  
    this.Label.value =_  
        "The temperature is " & Celsius(temp) &_  
        " degrees C. "  
End Sub
```

Function Procedures

A Function procedure is a series of VScript statements enclosed by the Function and End Function statements. A Function procedure is similar to a Sub procedure, but can also return a value. A Function procedure can take arguments (constants, variables, or expressions that are passed to it by a calling procedure). If a Function procedure has no arguments, its Function statement must include an empty set of parentheses. A Function returns a value by assigning a value to its name in one or more statements of the procedure. The return type of a Function is always a Variant.

In the following example, the Celsius function calculates degrees Celsius from degrees Fahrenheit.

```
Function Celsius(fDegrees)  
    Celsius = (fDegrees -32) * 5 / 9  
End Function
```

Getting Data into and out of Procedures

Each piece of data is passed into your procedures using an argument . Arguments serve as placeholders for the data you want to pass into your procedure. You can name your arguments any valid variable name. When you create a procedure using either the Sub statement or the Function statement, parentheses must be included after the name of the procedure. Any arguments are placed inside these parentheses, separated by commas. For example, in the following example, fDegrees is a placeholder for the value being passed into the Celsius function for conversion.

```
Function Celsius(fDegrees)  
    Celsius = (fDegrees -32) * 5 / 9  
End Function
```

To get data out of a procedure, you must use a Function. Remember, a Function procedure can return a value; a Sub procedure can't.

Using Sub and Function Procedures in Code

A Function in your code must always be used on the right side of a variable assignment or in an expression. For example:

```
| Temp = Celsius(fDegrees)
```

Or

```
| Message = "The temperature is " & Celsius(temp) &_  
|           " degrees C. "
```

To call a Sub procedure from another procedure, type the name of the procedure along with values for any required arguments, each separated by a comma. The Call statement is not required, but if you do use it, you must enclose any arguments in parentheses.

The following example shows two calls to the MyProc procedure. One uses the Call statement in the code; the other doesn't. Both do exactly the same thing.

```
| Call MyProc(firstarg, secondarg)  
| MyProc firstarg, secondarg
```

Notice that the parentheses are omitted in the call when the Call statement isn't used.

VScript Classes

As many hi-level languages VScript allow declaring and using classes. Classes may make easy to create, understand and support you code.

Define Classes and Create Instances

To declare class you can use statement class as in following example:

```
Class Person
    Dim Name, Sex, Birthday
End Class
```

Here we declare class Person with three class variables : Name, Sex, Age. To use this class we must create an instance:

```
Dim Andy
Set Andy = New Person
Andy.Name = "Andy"
Andy.Sex = 0
Andy.Birthday = #12-03-1982#
```

To create an instance we use keyword New and to assign new instance to the variable we use statement Set.

Using Class Procedures

Except for variables we can declare class procedures. For Example:

```
Class Person
    Dim Name, Sex, Birthday
    Sub ChangeName(value)
        This.LabelName.value = Name
        This.LabelAge.value = Age
    End Sub
    Function Age
        Age=DateDiff("yyyy", Birthday, Now)
    End Function
End Class
```

There example how to use these methods:

```
Andy.ChangeName "Andy Jr."
this.LabelAge.value = Andy.Age
```

Defining and Using Properties

Properties are special procedures that can get or set class variables in a special way. There are three types of class procedures that can describe property: Get, Let and Set. Property Get statement used to define get procedure of property. It's must return property value. Property Let and Property Set statements used to assign some value to property. Only one of these procedures may be declared at a

time. To declare read-only property we must only define Property Get procedure. To declare write-only property we must define only Property Let or Property Set. To define read/write property we must define both procedures. There is example:

```
Class Person
  Dim Name, Sex, Birthday
  Property Get SexString
    If Sex=0 Then
      SexString="Male"
    Else
      SexString="Female"
    End if
  End Property
  Property Let SexString(value)
    If value="Male" then
      Sex=0
    Else
      Sex=1
    End if
  End Property
End Class
```

Default Property

One property in a class may be default property. It's a property that will be returned while using class name in expressions. For example:

```
Class Person
  Dim Name, Sex, Birthday
  Default Property Get DefaultProperty
    DefaultProperty = Name
  End Property
End Class
```

There we declare class with default property DefaultProperty. Now when we use class name it will return Name of the person:

```
Dim Andy
Andy = new Person
Set Andy.Name = "Andy"
this.LabelName.value=Andy
```

VScript & VDOM Object / Server Objects

Main task of VScript is manipulating VDOM objects and their attributes. Each scripts are attached to a VDOM container and executed before the container rendering process.

Creating objects

VScript allow to dynamically create new VDOM Object during the execution of the Web application, this is very useful and flexible way to create dynamic content inside web application.

This mechanism is equivalent to create new controls in common desktop application.

To create new VDOM Objects you need to use a special embedded server object named server like this.

```
Dim myDynamicText
Set myDynamicText = server.CreateObject ("73a54f2e-4001-4676-93a0-804048a57081",this.ID)
```

Each VDOM object has a unique identifier, this one is needed to create the related VDOM object, you can find here a non exhaustive list of VDOM Object

Name	ID	Function
Bar	91a12281-c9a8-430a-8a2d-93903b4a264f	Create a colored rectangle
Breadcrumb	8a1650eb-17e5-4944-9e56-0a9817ce1665	Breadcrumb for navigation
Button	315381b8-f3f1-496c-92be-b65ebdd6b8a1	A picture with rollover and link to an other container
Calendar	b20c407a-2c83-4646-b64d-a82f5db26490	A calendar picker
Container	81d947af-1548-4a96-a1e0-d8a4c67c6ec2	A logical container to group objects
DbHtmlView	76f84567-ca9a-45de-9bbb-d5f13ea6a154	HTML feeded by a SQL request on DbSchema container
DbSchema	753ea72c-475d-4a29-96be-71c522ca2097	Container of relation database (Tables)
DbSimple	3a9827f6-0dd5-4239-9544-6e42ef0085ce	HTML feeded by a Table in a DbSchema container
DbTable	92269b6e-4b6b-4882-852f-f7ef0e89c079	VDOM Object that represent a Table
Debug	246a9164-487b-4f75-944b-2a6907b2b078	Output text for debugging information
Form	34f6ee59-9c50-4503-97c1-86c4e86bd1b7	Container to create a Form
FormButton	5be544cb-3d6b-4b75-ae79-b071fbe46094	A standard Form button
FormCheckBox	8077aa1c-6762-4719-a6ea-fdfb0bcfa0c2	A standard Form UI checkbox
FormDropDown	7029560a-ca17-4e32-a058-cf82b8facc33	A standard Form UI DropDown
FormPassword	6555559f-3092-49bd-8b91-ca15ba10a373	A standard Form Password field
FormRadioButton	213f1e8c-8a3e-452b-af33-4ca3139fe960	A standard Form UI RadioButton
FormRadioGroup	5ec776c5-23f6-4098-a69d-600b08b220b0	A standard Form UI RadioGroup
FormText	410ce9c6-5ae0-4c66-9c2b-80b7470e2927	A standard Form Text field
FormUploader	823833ac-0f63-431c-82e7-0a502af21c65	A special field to provide UI file navigation
HTML	7085bd26-e653-490b-908f-61208c260a86	To inject direct HTML in the container
Image	0d36c35d-9508-440f-bfec-668f3db8cfeb	Show a picture
Menu	03741d38-c9f3-4526-acb9-71c7aa00b3b2	Create a set of button according to the application tree struct.
RichText	82a69b02-9fba-47d0-b206-6fd1769b0ebd	Text with advanced WYSIWYG functions
SensitiveZone	7b39c919-de7f-4b77-b048-aae8bcf8edf5	Transparent rectangle clickable.
Table	4858cfb6-735e-47be-b500-d63720fc4119	Table container
TableCell	3de32e4a-1493-49c3-add7-ddf8738e1530	Table Cell container
TableRow	19a2a656-40f1-43ca-9eba-eb55d033b1d4	Table Row container
Text	73a54f2e-4001-4676-93a0-804048a57081	Simple text with uniform attribute like font, size, ...
Time	1052fb85-22db-40e9-a4e1-5b1e1a3b2280	Show the current time.

Using objects

When scripts are executed container that owns this script map it to a special object named "this". It allows access to container's objects and attributes, etc...

All attribute of each object (container & terminal objects) are available thru a standard dot notation.

This example shows how to get title attribute value of the current container:

```
| Dim Title  
| Title = this.Title
```

To set the value to this attribute we write:

```
| this.Title = "New Title"
```

The access of any child object contain in the current container use also the dot notation like this:

```
| this.childObject.color = "FFFFFF"
```

Get access to foreign objects

It could be needed to be able to modify or interact with object outside the current container, to perform this action you needed to use a special method on server object.

```
| Dim Label  
| Label = server.GetObject("7076ef50-ddb2-4809-9741-  
| 1389454ab64e")
```

This number 7076ef50-ddb2-4809-9741-1389454ab64e is the unique identifier of the object inside the internal server VDOM Memory.

Embedded objects

Server

The Server object is designed to access to the methods and properties of the server. Most of these methods and properties serve as utility functions. These methods are divided into two groups: objects manipulation, we have already discussed of them and auxiliary functions. These one are HTML Encode and URL Encode.

The HTML Encode method applies HTML encoding to a specified string. This is useful to encode form data and other client request before using it in your Web application. Encoding data converts potentially unsafe characters to their HTML-encoded equivalent.

If the string to be encoded is not DBCS, HTML Encode converts characters as follows:

- The less-than character (<) is converted to <.
- The greater-than character (>) is converted to >.
- The ampersand character (&) is converted to &.
- The double-quote character (") is converted to ".

Any ASCII code character with a value greater-than or equal to 0x80 is converted to &#<number>, where <number> is the ASCII character value.

The URL Encode method applies URL encoding rules, including escape characters, to a specified string.

URL Encode converts characters as follows:

- Spaces () are converted to plus signs (+).
- Non-alphanumeric characters are escaped to their hexadecimal representation.

RegExp

Regular expression (RegExp) object provides simple regular expression support. The following code illustrates the use of RegExp object.


```
Dim regex, match, matches, result
Set regex=new regexp
regex.pattern="w."
regex.ignorecase=true
regex.global=true
set matches=regex.execute("w1 w2 w3")
for each match in matches
    result=result & "position " & match.firstindex
    result=result & " value " & match.value & ";"
next
```

RegEx object use usual regular expression syntax and has three property: Global, IgnoreCase and Pattern.

Global property sets or returns a Boolean value that indicates if a pattern should match all occurrences in an entire search string or just the first one.

IgnoreCase property sets or returns a Boolean value that indicates if a pattern search is case-sensitive or not.

And Pattern property sets or returns the regular expression pattern being searched for.

Besides RegEx object has three method: Execute, Replace and Test.

Execute method executes a regular expression search against a specified string as in previous example and has one argument - string upon which the regular expression is executed. The Execute method returns a Matches collection containing a Match object for each match found in string. Execute returns an empty Matches collection if no match is found.

Replace method replaces text found in a regular expression search. Its has two arguments: first string is the text string in which the text replacement is to occur and second string is the replacement text string. There is an example:

```
| result=regex.replace("w1 w2 w3","w")
```

In this example we replace character pairs starts with "w" to character "w" .

And method Test executes a regular expression search against a specified string and returns a Boolean value that indicates if a pattern match was found. Its has only one argument - string upon which the regular expression is executed. Example:

```
| result=regex.test("w1")
```

VScript HTTP Objects

To support programmer we emulate some standard ASP objects that facilitate writing scripts. There are four objects: Server, Request, Response and Session.

Request

The Request object retrieves the values that the client browser passed to the server during an HTTP request. The Request object defines the following properties: Cookies, Form, QueryString, ServerVariables

All variables can be accessed directly by calling Request(variable) without the collection name. In this case, the Web server searches the collections in the following order:

- QueryString
- Form
- Cookies
- ServerVariables

If a variable with the same name exists in more than one collection, the Request object returns the first instance that the object encounters.

Example of using request object:

```
For each name in request.servervariables
    Result=result & name & "-"&_
    Request.servervariables(name)
Next
```

The Cookies collection enables you to retrieve the values of the cookies sent in an HTTP request.

The Form collection retrieves the values of form elements posted to the HTTP request body, with a form using the POST method.

The QueryString collection retrieves the values of the variables in the HTTP query string. The HTTP query string is specified by the values following the question mark (?). Query strings are also generated by sending a form or by a user typing a query into the address box of the browser.

The ServerVariables collection retrieves the values of predetermined environment variables and request header information.

Server variables obtain most of their information from headers. It is wise to not trust the data that is contained in headers, as this information can be falsified by malicious users. For example, do not rely on data such as cookies to securely identify a user.

Response

You can use the Response object to send output to the client. This object contains AddHeader and Redirect methods and Cookies collection.

The AddHeader method adds a new HTML header and value to the response sent to the client. It does not replace an existing header of the same name. After a header has been added, it cannot be removed.

The Redirect method causes the browser to redirect the client to a different URL.

The Cookies collection sets the value of a cookie. If the specified cookie does not exist, it is created. If the cookie exists, it takes the new value, and the old value is discarded.

The IsClientConnected property is a read-only property that indicates if the client has reset the connection to the server.

Session

You can use the Session object to store information needed for a particular user session. Variables stored in the Session object are not discarded when the user jumps between pages in the application; instead, these variables persist for the entire user session.

The Web server automatically creates a Session object when a Web page from the application is requested by a user who does not already have a session. The server destroys the Session object when the session expires or is abandoned.

Methods / Property of the session objects

- Abandon : Destroy the current session of the user

```
Session.Abandon
// Not implemented in server ver 0.9.2930
```

- Timeout : allow to set or read the current session timeout

```
Session.Timeout = 100000 `value in millisecond
Or
Session.Timeout = Session.Timeout
// Not implemented in server ver 0.9.2930
```

- SessionId : (ReadOnly) get the current identifier of the session

```
ID = Session.SessionId
```

- Variables

```
`Store a variable
Session.Variables("MyVarName") = MyVar
```

```
'Retrieve a variable  
MyVar = Session.Variables("MyVarName")
```

The Timeout property specifies the time-out period assigned to the Session object for the application, in minutes. If the user does not refresh or request a page within the time-out period, the session ends.

VScript & VDOM DB Acces

General

The VDOM server provide a complete DB solution based on SQLite. But the user don't manage directly the DB, it use the VDOM Object model for it.

A data base in VDOM is created upon a DBSchema container, this container contain DB Table Object that represent a standard relational database.

Some object can directly use this Db model like simple Db or DB HTML View, but all application can't use directly this direct connection and need some script to manipulate data.

VScript provide a complete acces to DB thru different embedded object.

Connection Object

Open a connection

The first operation to perform is to connect to the DB, as we explained upper a DB for VDOM is an object, the connection is made like this:

```
Dim connection
Set connection = new vdomDbConnection
Connection.open("mydb")
```

`vdomdbconnection` is the embedded object used to connect to the DB

it has one method `open` with the name of the Db Schema in agrument.

Close a connection

To free resources and safely terminate DB operation a connection has to be closed with the method `close`.

```
| Connection.close
```

Query the DB

A query can or not return data, a SELECT query will return a amount of data stored into a special object named recordset, but the UPDATE & DELETE query will return nothing and can be perform with another method. Those two case have their own method to be executed, the first one use **query** and the second **execute**.

Query that return data

```
Dim connection,recordeset,record
Set connection = new vdomDbConnection
Connection.open("mydb")
Set recordset=connection.query("SELECT * FROM myTable")

this.myText.value = ""

for each record in recordset
    this.myText.value = this.myText.value +_
        "Champ0:" + cstr(record(0)) + "<br>"
    this.myText.value = this.myText.value +_
        "Champ1:" + cstr(record(1)) + "<br>"
    this.myText.value = this.myText.value +_
        "Champ2:" + cstr(record(2)) + "<br>"
    this.myText.value = this.myText.value +_
        "-----"
next

connection.close
```

Query that return no data

```
Dim connection
Set connection = new vdomDbConnection
Connection.open("mydb")
connection.execute("INSERT INTO myTable_
    VALUES '1','2' ")
connection.close
```

VScript

CookBook

Page access control

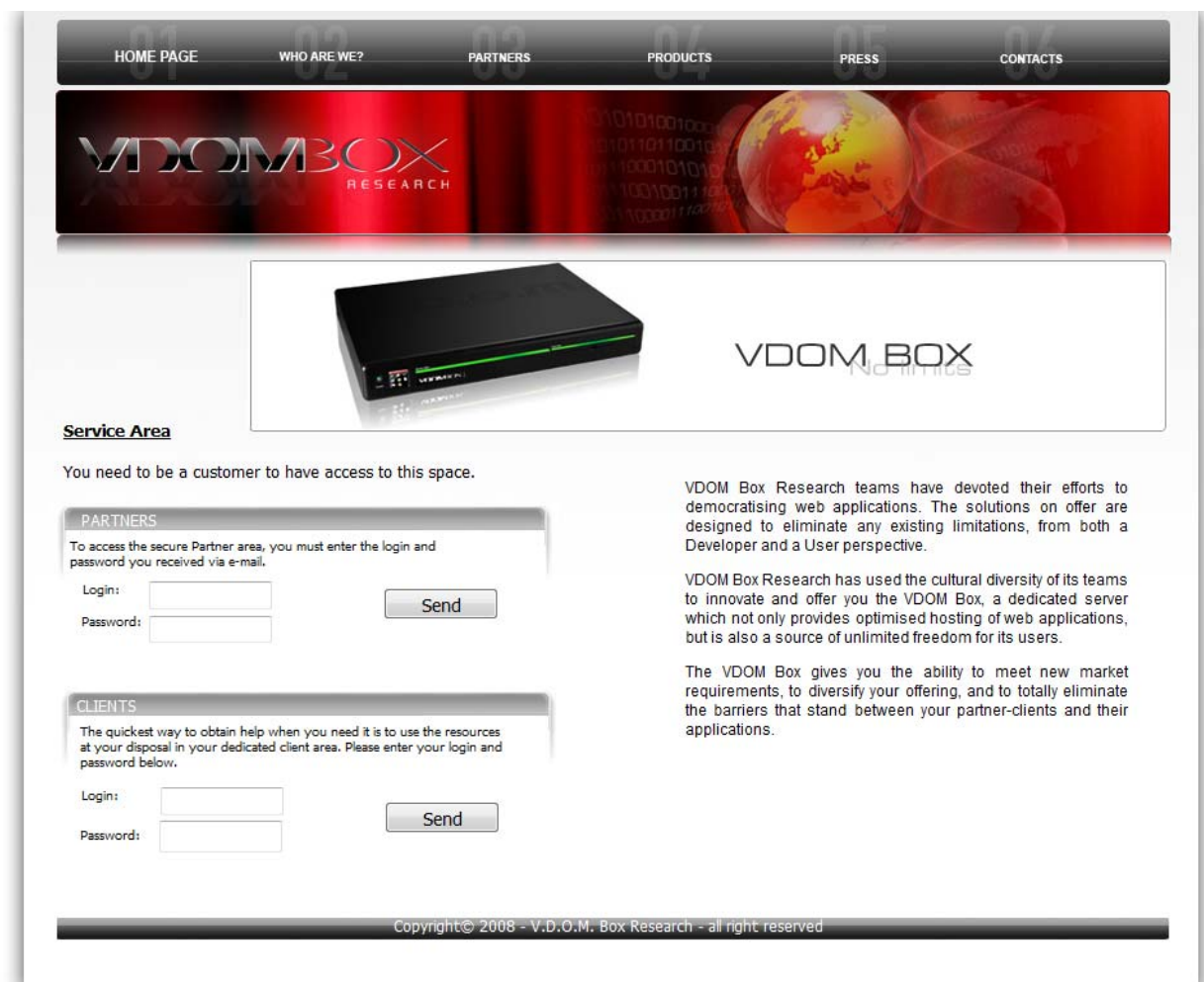
General

The VDOM HTML Container provide a easy solution to control the access to container with two simples properties.

This example will show you how connect the user control access with simple script.

Application

The application is a simple website with to secure space, to enter in the secure space you need to log in, if the user login & password is wrong you will be redirect to an access denied page (figure 2), all the user information are stored into the VDOM DB.



The screenshot displays the first page of the VDOM Box Research website. At the top, a dark navigation bar contains links for HOME PAGE, WHO ARE WE?, PARTNERS, PRODUCTS, PRESS, and CONTACTS. Below this is a red banner with the VDOM BOX RESEARCH logo and a globe graphic. The main content area features a black server unit with a green light bar and the text "VDOM BOX NO LIMITS".

Service Area

You need to be a customer to have access to this space.

PARTNERS

To access the secure Partner area, you must enter the login and password you received via e-mail.

Login:

Password:

CLIENTS

The quickest way to obtain help when you need it is to use the resources at your disposal in your dedicated client area. Please enter your login and password below.

Login:

Password:

VDOM Box Research teams have devoted their efforts to democratising web applications. The solutions on offer are designed to eliminate any existing limitations, from both a Developer and a User perspective.

VDOM Box Research has used the cultural diversity of its teams to innovate and offer you the VDOM Box, a dedicated server which not only provides optimised hosting of web applications, but is also a source of unlimited freedom for its users.

The VDOM Box gives you the ability to meet new market requirements, to diversify your offering, and to totally eliminate the barriers that stand between your partner-clients and their applications.

Copyright© 2008 - V.D.O.M. Box Research - all right reserved

Figure 1: application first page

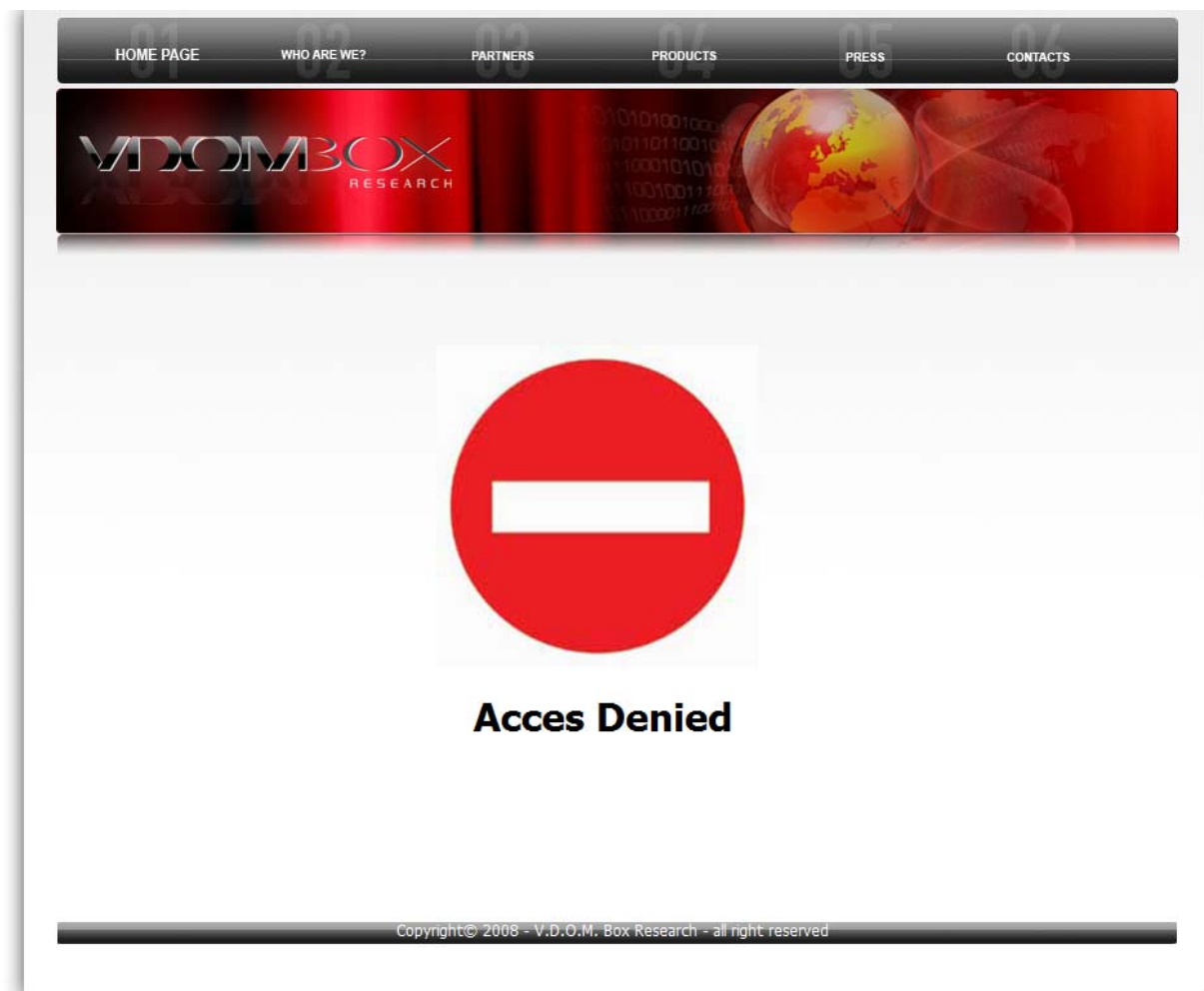


Figure 2: Access denied page

 VDOM XML (Figure 1)

```
<HTMLCONTAINER name="Home_Page" description="First page of the web site" title="Home Page">
  <IMAGE name="imgServiceAerea" top="383" value="60f5ffe9-d35c-4299-a7e4-4a3be2d84c73" height="129" zindex="1" width="397" left="48" />
  <IMAGE name="bandeau" value="e55cecbf-a92f-40a6-8134-ab2322e2cb69" height="768" width="973" />
  <FLASHANIMATION name="FlashBox" top="195" height="134" zindex="5" width="720" file="f993eb27-3773-49ff-a115-a1217aeec0a" left="202" />
  <RICTEXT name="txtInfpService" top="320" zindex="1" width="372" left="58">
    <Attribute Name="value"><![CDATA[<span style="font-size: small" class="Apple-style-span"><span style="font-weight: bold" class="Apple-style-span"><span
style="font-family: tahoma, arial, helvetica, sans-serif" class="Apple-style-span"><span style="text-decoration: underline" class="Apple-style-span">Service
Area</span></span></span></span></div><span style="font-family: tahoma; font-size: 13px" class="Apple-style-span"><br /></span></div><div style="text-align:
justify"><span style="font-family: tahoma" class="Apple-style-span"><span style="font-size: small" class="Apple-style-span">You need to be a customer to have access
to this space.&nbsp;</span></span></div>]]></Attribute>
  </RICTEXT>
  <RICTEXT name="vdom_box" top="353" zindex="1" width="346" left="544">
    <Attribute Name="value"><![CDATA[<p align="justify"><span><font><font><span style="font-family: arial, helvetica, sans-serif" class="Apple-style-span">VDDM Box
Research teams have devoted their efforts to democratising web applications. The solutions on offer are designed to eliminate any existing limitations, from both a
Developer and a User perspective.</span></font></font></span></p><p align="justify"><span><font><span style="font-family: arial, helvetica, sans-serif"
class="Apple-style-span">VDDM Box&nbsp;&nbsp;&nbsp;Research&nbsp;&nbsp;&nbsp;has used the cultural diversity of its teams to innovate and offer you the VDDM Box, a dedicated server which
not only provides optimised hosting of web applications, but is also a source of unlimited freedom for its users.</span></font></span></p></p>]]></Attribute>
  </RICTEXT>
</HTMLCONTAINER>
```

```

align="justify"><font><font><span style="font-family: arial, helvetica, sans-serif" class="Apple-style-span">The VDOM Box gives you the ability to meet new market
requirements. to diversify your offering, and to totally eliminate the barriers that stand between your partner-clients and their
applications.</span></font></font></p></Attribute>
</RICHTEXT>
<TEXT name="txtCopyright" top="708" value="Copyright© 2008 - V.D.O.M. Box Research - all right reserved" zIndex="5" width="860" color="FFFFFF" align="center"
left="54"/>
<FORM name="formPartner" target="299ac020-341f-441c-9495-6c68f01a1b51" top="412" height="95" zIndex="1" width="367" left="63">
  <TEXT name="textInfoPartner" value="To access the secure Partner area, you must enter the login and password you received via e-mail." width="331"
fontSize="10"/>
  <TEXT name="textPartnerLogin" top="34" value="Login:" width="161" fontSize="10" left="10"/>
  <TEXT name="textPartnerPass" top="59" value="Password:" width="100" fontSize="10" left="9"/>
  <FORMBUTTON name="formbutton_99c7bccac868_43fc_b792_8b93afb2d54b" top="39" left="245"/>
  <FORMPASSWORD name="partpassword" top="61" height="21" width="96" left="62"/>
  <FORMTEXT name="partlogin" top="33" width="96" left="62"/>
</FORM>
<FORM name="formClient" top="555" height="115" zIndex="3" width="360" left="66">
  <TEXT name="text_36a40c2e_6fe8_45ad_ba6a_2bbb9527308b" top="2" zIndex="5" width="334" fontSize="10" left="5">
    <Attribute Name="value"><![CDATA[The quickest way to obtain help when you need it is to use the resources at your disposal in your dedicated client area.
Please enter your login and password below.]]></Attribute>
  </TEXT>
  <TEXT name="text_16891cdd_862d_4f08_b786_3b781b773d64" top="52" value="Login:" width="52" fontSize="10" left="6"/>
  <FORMBUTTON name="formbutton_a954f321_8e22_44d2_b7b3_facbb56c3289" top="63" left="243"/>
  <FORMTEXT name="formtext_62f21479_9715_4a57_b53d_54890468c6fd" top="51" zIndex="5" width="96" left="68"/>
  <TEXT name="text_4af0fa34_5587_44fa_9055_fb221d14405f" top="83" value="Password:" width="63" fontSize="10" left="5"/>
  <FORMPASSWORD name="formpassword_ee710a50_b5f7_4f78_8791_88bdc6cfc59" top="78" width="96" left="67"/>
</FORM>
<IMAGE name="imgFond" top="709" value="e62bc819-fd54-40f1-aec4-1d3aeb94e6" height="17" zIndex="1" width="870" left="53"/>
<IMAGE name="imgClientAerea" top="528" value="60f5ffe9-d35c-4299-a7e4-4a3be2d84c73" height="129" zIndex="1" width="397" left="51"/>
<RICHTEXT name="Partners" top="391" zIndex="5" left="72">
  <Attribute Name="value"><![CDATA[<span style="color: #ffffff" class="Apple-style-span">PARTNERS</span>]]></Attribute>
</RICHTEXT>
<RICHTEXT name="CLIENT" top="536" zIndex="5" left="68">
  <Attribute Name="value"><![CDATA[<span style="color: #ffffff" class="Apple-style-span">CLIENTS</span>]]></Attribute>
</RICHTEXT>
<TEXT name="userloggedpart" visible="0" top="432" value="User _____ Logged" zIndex="5" width="367" align="center" left="60"/>
<SENSITIVE name="sensitive_c561e96f_1b00_4a68_b69f_e1eb5ef4c1fd" containerlink="299ac020-341f-441c-9495-6c68f01a1b51" top="7" zIndex="5" width="74"
left="361"/>
</HTMLCONTAINER>

```

In this code two objects are important FROM container and TEXT, by script we are going to modify some of their values to take care about the current login state.

```

1 select case cstr(Session.variables("loggedon"))
2     case ""
3         this.userloggedpart.visible=0
4     case "partner"
5         this.formPartner.visible=0
6         this.userloggedpart.visible=1
7         this.userloggedpart.value="User_
8             "+Session.variables("useridentity")+ "_
9             logged"
10    case "client"
11        this.formClient.visible=0
12
13 end select

```

In this VScript code we can see the acces to the VDOM object of the current container by this.objectName.attribut.

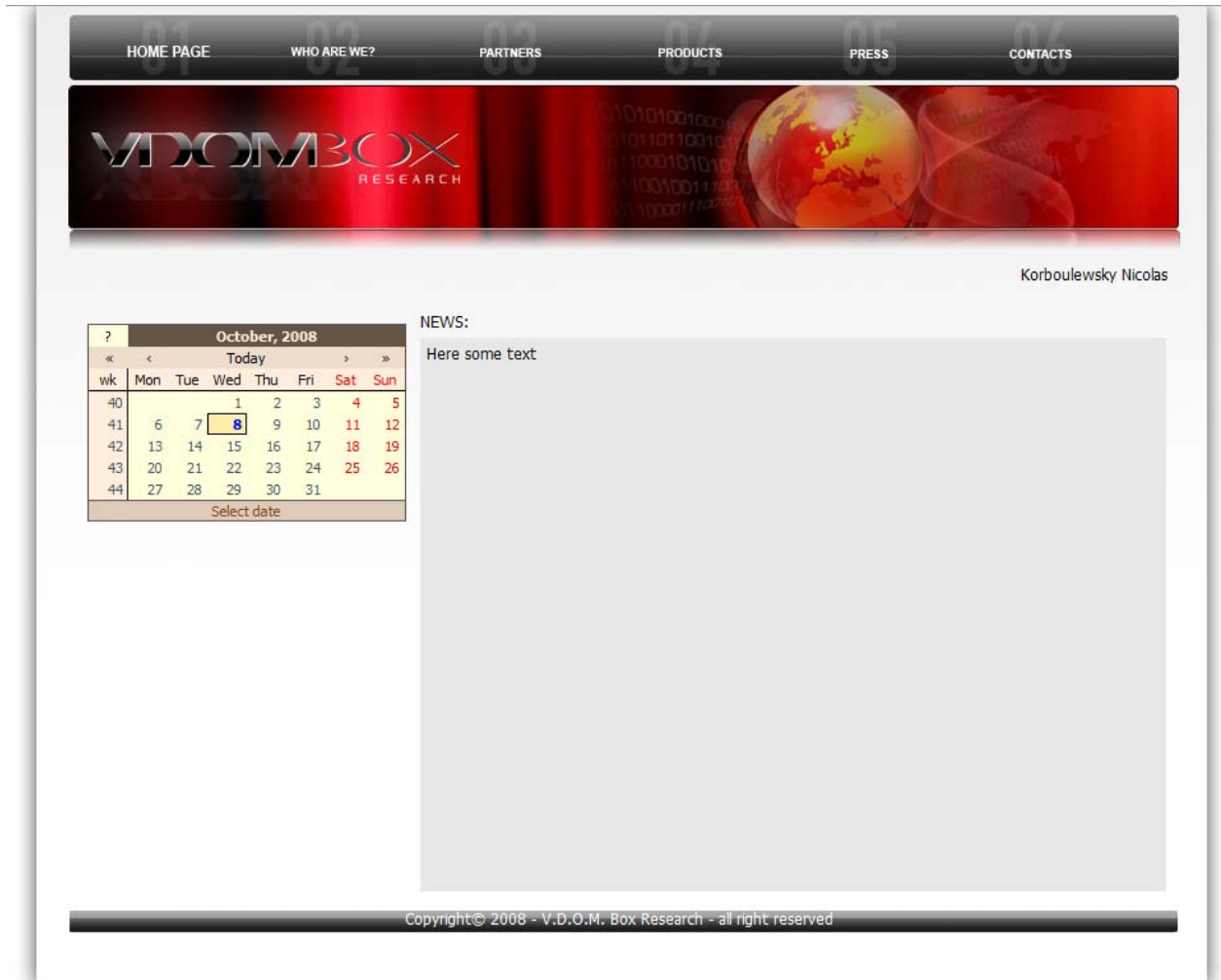


Figure 3: Access ok

```

1 dim connection,SecurityData
2 Set connection = new VdomDbConnection
3
4     if cstr(Session.variables("SecurityCode"))="" then
5         if request.form("partlogin")<>"" and_
6             request.form("partpassword")<>"" then
7             connection.open("applidb")
8
9             Set SecurityData = connection.query("SELECT_
10                SecurityCode,name,surname FROM user WHERE_
11                Login='"+request.form("partlogin")+"' AND_
12                Password='"+request.form("partpassword")+"'")
13
14             For each record in SecurityData
15                 Session.variables("SecurityCode")=record(0)
16             Next
17             Session.variables("useridentity") = record(1)+_
18                 " "+record(2)

```

```

13         this.useridentity.value =_
           Session.variables("useridentity")
14         Session.variables("loggedon")="partner"
15         connection.close
16     else
17         wscript.echo "No login and/or password"
18     end if
19 else
20     this.useridentity.value =_
           Session.variables("useridentity")
21 end if

```

VDOM XML (Figure 3)

```

<HTMLCONTAINER name="FormPartner" description="Partner space" title="Welcome Partner" securitycode="gq13Zdl" deniedlink="a9de86aa-3253-42ef-86d9-
f242f5de6439">
  <COPY name="copyBandeau" source_object_cache="2923b1c0-9e28-4f8e-b316-ed67520012cf" zindex="0" source_object="2923b1c0-9e28-4f8e-b316-
ed67520012cf"/>
  <COPY name="copyLgn" top="664" source_object_cache="18a573b4-2334-4a77-b3c1-1b54d46809e4" zindex="0" source_object="18a573b4-2334-4a77-b3c1-
1b54d46809e4" left="20"/>
  <COPY name="copyCopyright" source_object_cache="8b22639a-77d2-4663-853b-5a9837e31960" zindex="0" source_object="8b22639a-77d2-4663-853b-
5a9837e31960"/>
  <SENSITIVE name="sensitive_8a43010b_ea6a_4bb8_bf49_5c1734467137" containerlink="536553fc-aa28-4a52-8371-481578ab05e8" top="7" zindex="10" width="103"
left="77"/>
  <TEXT name="useridentity" top="203" value="Welcome name surname" zindex="10" align="right" left="514"/>
  <COPY name="Copyfond" top="709" source_object_cache="f854025c-7722-48de-be38-561596006ddc" zindex="10" source_object="f854025c-7722-48de-be38-
561596006ddc" left="53"/>
  <TEXT name="text_d173cb5c_2d30_4b6c_84f1_65cd1326cc8f" top="240" value="NEWS:" zindex="10" left="328"/>
  <BAR name="bar_93bc8cce_6adb_4c3e_9bel_b2b2a6058906" top="260" height="434" zindex="10" width="585" color="E8E8E8" left="328"/>
  <RICHTEXT name="richtext_ae642ef2_2614_4c03_bda8_1623dbel7974" top="265" value="Here some text" zindex="10" width="573" left="333"/>
  <FORM name="sendcalendar" target="299ac020-341f-441c-9495-6c68f01alb51" top="242" height="189" zindex="10" width="266" left="60">
    <CALENDAR name="calendar" todayColor="ffeeaa" skin="/ae4174eb-9a79-412e-815d-e7186d603935.res" borderColor="665555" top="7" titleFontColor="ffeedd"
fontSize="11" dayNameColor="ffeedd" weekColor="ffeedd" headFontColor="000000" dateAreaColor="ffffdd" navigationPanelColor="eeddcc" tipColor="ddccbb"
tipFontColor="884400" headColor="eeddcc" todayFontColor="0000ff" weekendColor="ff0000" titleColor="665544" left="7"/>
  </FORM>
</HTMLCONTAINER>

```

The security access page is based on an attribute of HTML container name securitycode as shown upper, if a special session variable (Session.variables("SecurityCode")) is equal to one of the code saved in this attribute the page will be shown if not it will be redirect to the container provided in the deniedlink attribute.

The upper code shows how we get this value from the database and save it to the session variable. No more code is needed to provide a complete security access management.