



10

WebBoard Open Scripting

WebBoard's object model scripting feature—called Open Scripting—gives you access to WebBoard's inner workings. By using WebBoard object methods and properties, you can write scripts that replace or alter built-in WebBoard operations in boards and conferences that use the frames interface. For example, you can supply scripts for your own user authentication or process form data before it gets to WebBoard. Scripts let you customize much more than WebBoard's appearance, building a site that works for you—and your users.

You don't have to learn a new language to use Open Scripting; simply use the scripting language you're most comfortable with. Assuming you've got the scripting engine (and it's compatible with the Windows Scripting Host), WebBoard runs the script, identifying the engine to use by your script's file extension. WebBoard is configured to recognize two common scripting languages, VBScript and JavaScript; you make a single Registry edit to add others.

This chapter describes the three types of scripts WebBoard supports—form scripts, tag scripts, and authentication scripts. You'll learn what you can accomplish with each type, requirements for writing the script, and how to get WebBoard to run it. For each type of script, there are a few sample scripts you can use as templates in creating your own. As WebBoard users create scripts, a script repository on WebBoard Central allows you to share them and benefit from each others' ideas.

A reference to the WebBoard object methods and properties follows the information on how scripting works and how to write scripts.

Open Scripting Basics

WebBoard's object-based Open Scripting exposes WebBoard's internal information and some WebBoard features to scripts you write in your choice of scripting language. Scripts can perform a variety of tasks, such as overriding basic or cookie authentication with your own authentication routine, or checking user profile information for accuracy (missing fields, state abbreviations, and so on) and correcting it where necessary. You can also use scripts to gain access to administrative features such as user paging and board and systemwide broadcasting. Another possible application is to use a script to display random or sequential advertisements.

Note

WebBoard Open Scripting works only with WebBoard's frames interface.

The WebBoard object methods and properties available to scripts provide information about the board, the users, and the web server, and allow you to manipulate many aspects of the board and user input, including:

- The Authentication object lets you authenticate and add users.
- The Board object makes available properties of the current board, such as the board alias, colors, name, manager, and more.
- The Filesystem object gives your script access to basic file manipulation routines.
- The Form object lets your script read fields from a post and, under some conditions, set new values.
- The Request object provides information such as basic authentication information, hostname, user agent, IP address, cookies, referer, and other details about the HTTP request.
- The Response object allows the script to send HTML back to the browser.
- The Server object provides information on server properties, for example, product name and version, the WebBoard administrator's email address, and the HTTP port. It also lets you page boards, send email, and log messages to *Webboard.log*.
- The User object makes available user properties, such as the current user's name, email, location, and all other settings on the user profile. In addition, you can change certain properties, page the user, and more.

Scripting Engines Supported

WebBoard makes it easy for you to choose the scripting engine you prefer (and allows for the possibility of new ones being developed). Open Scripting works with any scripting engine that is compatible with the Windows Scripting Host. The extension of your script tells WebBoard which scripting engine to use. When WebBoard encounters the <WB-SCRIPT> tag, it looks in the following Registry key to see which engine matches the filename extension of the script:

```
HKEY_LOCAL_MACHINE\SOFTWARE\O'REILLY\WEBBOARD\3.0\SCRIPTING
```

By default, WebBoard creates two name/value pairs in the Registry key: JS/JavaScript and VBS/VBScript. If you want to use a different scripting engine, install it and then add a name/value pair to the *Scripting* key, where the name is the extension and the value is the scripting engine.

Types of Scripts

WebBoard's Open Scripting supports three types of scripts—form scripts, tag scripts, and authentication scripts. The following sections describe each type of script and how to run it. Sample scripts in each category are provided in the *WebBoard/Scripts* folder. These sample scripts demonstrate how to write a script for a specific purpose. Use the sample scripts as templates for creating your own scripts; they are not for actual use.

Warning

Always back up your WebBoard database, your HTML files, and scripts before using scripts that may alter your data.

Form scripts

Form scripts let you process form field information sent to WebBoard by specific forms before WebBoard processes it. For example, you can change or add data on the user profile form or the login form. Form scripts can process messages before they are posted and can access JavaScript chat messages. Furthermore, you can access any type of data source from your script, enabling such features as expanding acronyms or adding extra information about the user posting the message. You can also cancel the post operation, replacing it with your own message. Here's a list of possible uses for form scripts:

IP Blocking

Create a login script to disallow (or only allow) logins by those at a certain IP address (or block of IP addresses).

Filth Filter

Create your own “filth filter” to change (or remove) foul language from messages and chat posts (only JavaScript chat).

Make Corrections

Alter the contents of any field on the form. For example, make sure the letters in the state field of the form on *USEREDIT.HTML* are in uppercase by a simple script action such as:

```
request.SetFormValue "state", UCase(user.State)
```

Cancel current operation

You can use a form script to cancel the current operation. For example, don’t allow the user to log in after checking the IP address.

Form scripts must be named according to a list of preset names and placed in the board’s HTML subfolder. WebBoard automatically runs the script at the proper time. Table 10-1 shows the forms you can alter and the script name to use. The script filenames are shown without an extension. You give the proper extension to match your scripting engine, as explained above in “Scripting Engines Supported.” For example, *NEWUSER.** becomes *NEWUSER.VBS* or *NEWUSER.JS*, depending on the configured script engine. Put *NEWUSER.VBS* in the *WebBoard/HTML/Faculty* folder, and every time the New User form is submitted on the *Faculty* board, WebBoard runs the script before processing the data.

Table 10-1 Script to use to alter a given form

Form (HTML files)	Script name
New User (<i>newuser.html</i>)	<i>NEWUSER.*</i>
Edit User (<i>useredit.html</i>)	<i>USEREDIT.*</i>
Message Save (<i>postmsg.html</i> , <i>postmsg-f.html</i> , <i>preview.html</i>)	<i>POSTMSG.*</i>
Login (<i>login.html</i> , <i>loginfailed.html</i>)	<i>LOGIN.*</i>
JavaScript Chat Post (<i>Chatbar.html</i> , <i>Chatcreate.html</i>)	<i>POSTCHAT.*</i>

WebBoard comes with two sample form scripts, described below. These scripts are for demonstration purposes only; create your own scripts to run on your site. Remove the number from the script name in order to run it. The number enables you to store multiple scripts of the same type.

POSTMSG1.VBS

The *POSTMSG1.VBS* sample script replaces common acronyms in messages with whole words. For example, “WWW” becomes “World Wide Web.” The HTML pages affected are *POSTMSG.HTML*, *POSTMSG-F.HTML*, and *PREVIEW.HTML*.

LOGIN1.VBS

The *LOGIN1.VBS* sample script cancels a login based on conditions you define. The script gives examples of denying access based on a login name, IP address, or block of IP addresses. The HTML pages affected are *LOGIN.HTML* and *LOGINFAILED.HTML*.

The following special methods are available only to form-preprocessing scripts. See the object and method documentation for additional details.

- `request.SetFormValue`
- `response.SendCancelMsg`
- `response.SendCancelMsgRaw`

Tag scripts

Tag scripts, or WB-SCRIPT scripts, perform an action or return HTML. You can change user information, display HTML on the fly, send email, and page users and boards. You run tag scripts from the WebBoard HTML pages, using the special WebBoard tag, `<WB-SCRIPT>`. When WebBoard encounters the tag, it looks for the script and runs it before the page is sent to the browser. If the script has output, the output replaces the tag. There is no limit to the number of `<WB-SCRIPT>` tags you can place on a page.

The syntax for the `<WB-SCRIPT>` tag is simple:

```
<WB-SCRIPT file="script_name">
```

WebBoard looks for the script in the HTML subfolder for the board. If it doesn't find it there, it looks in WebBoard's root HTML folder. You can also specify a full path or a UNC path. For example:

```
<WB-SCRIPT file="C:\foo\bar\script_name">
```

Insert the `<WB-SCRIPT>` tag on the HTML page where you want the script to run. Any script output replaces the tag when the page is sent to the browser. For example, you could write a script that checks for special days and greets the user appropriately on the *INTRO.HTML* page. The tag:

```
<WB-SCRIPT file="intro.vbs">
```

becomes the message:

```
Happy Valentine's Day, Ramon.
```

The sample script *GOODMORNING.VBS* demonstrates how to replace a `<WB-SCRIPT>` tag in an HTML page with a greeting to the user that changes depending on the time of day:

```

Sub Main()
    Dim h

    h = Hour(Now)

    If h >= 0 AND h < 12 Then
        response.send "<p>Good morning, " & user.FirstName & "!"
    Else
        If h >= 12 AND h < 18 Then
            response.send "<p>Good afternoon, " & user.FirstName &
"!"
        Else
            If h >= 18 AND h <= 24 Then
                response.send "<p>Good evening, " & user.
                    FirstName & "!"
            End If
        End If
    End If
End Sub

```

Authentication scripts

Authentication scripts override the built-in WebBoard authentication methods. Writing your own authentication scripts, you can authenticate via an NT user database or via a DSN to any ODBC data source. Your authentication script can check to see if a user is already in the WebBoard user database and approve or fail the authentication. The Authentication object methods expose many properties on the user profile, so you can add new users and make settings for users with your authentication script as well.

Authentication script names must be named *auth*. As described earlier, the file extension indicates which script engine to use to run the script, for example, *auth.pls* or *auth.vbs*. For storage, you can distinguish files by adding optional names after *auth*, such as *-file*. But before running it, remove the extra text from the filename. Place the script in a board's HTML subfolder or in the main HTML folder, and it will be run in place of WebBoard's own authentication scheme. The following sample authentication scripts are included with WebBoard:

AUTH-WBDB.VBS

This simple script demonstrates how to perform authentication using WebBoard's own database. While not all that useful, it is a good place to start learning how to replace WebBoard's internal authentication with your own.

AUTH-FILE.VBS

The *AUTH-FILE.VBS* sample script authenticates using a text file, where each username and password is on a single line, delimited by a comma.

AUTH-ODBC.VBS

You can authenticate using any ODBC data source on any platform. The *AUTH-ODBC.VBS* sample script performs authentication using a user database from any ODBC-compatible data source. Just create a DSN, modify a few settings, and you're on your way.

AUTH-NT.VBS

The *AUTH-NT.VBS* sample script lets you authenticate users with an NT user database. It requires Microsoft's Active Directory Services Interfaces 2.0. If you choose to use NT-based WebBoard authentication, remember to take your overall security needs into account in regard to sending unencrypted NT usernames and passwords over the Internet.

Case Study: Solving Problems with Scripts

A technical support team uses WebBoard for frontline technical support. Each of the company's products has its own board and the boards have conferences for such areas as installation questions, user-to-user support, and bug reports. For bug reports, it's necessary to collect very specific information, not only about the problem encountered, but about the configuration of the user's system. "Read Me First" topics with instructions on how to file a bug report were not sufficient, so the technical support team decides to tackle the problem with scripts.

Writing and implementing a script changes the default message posting form to a form that collects the needed information. Now, when a user clicks Post for a blank message form, WebBoard encounters a <WB-SCRIPT> tag as it reads the *postmsg.html* file (the HTML file for message forms). Before displaying the message form in the user's browser, WebBoard runs the tag script. The script places specific fields for collecting information—product version, operating system, error message, and so on—on the message form. It also populates the drop-down menus for these fields with appropriate choices based on the product for which this board is used. The information is stored in a database, which the script consults while running. This added feature—the database—makes it possible for the tech support team to use the script on all product support boards, customizing the form for the specific product.

After the user fills out the form and posts it, a form script processes the message. The script is named *postmsg.**, which identifies it to WebBoard as an Open Scripting form script for processing the data on a message form. The script formats the information from the various fields so the message reads easily and even adds a bug ID number before the topic. The script also stores the field data from the message form in a database so that the technical support team can use it, for example, with a statistics generator.

You may not need a bug report form, but with this example as a jumping off place, you can probably think of appropriate features to add to your own boards

with a clever combination of scripts. Do it yourself, or call in someone with more experience with writing scripts, but by all means experiment with Open Scripting to tailor your WebBoard site to work for you and your users.

Object Reference

The object reference gives the syntax for each object's methods and properties, describes the return values, and gives some code examples. All the code examples are written in VBScript.

Auth Object

The Auth object lets you authenticate and add users.

Methods

In the syntax, parameters in italics are optional.

AddUser(*loginName*, *password*, *email*, *firstName*, *lastName*, *city*, *state*, *country*, *homePage*, *bio*, *signature*, *frames*, *fullView*, *paging*, *reverseTopics*, *hideEmail*, *autoMarkNew*)

Adds a user to the current board, and makes the new user available as an active user (visible in user listings and so on). The minimum required parameters for this method are `loginName` and `password`. The optional parameters relate to fields on the user profile (to see the profile, choose Edit User Profile from the More Options menu).

Before using `AddUser`, make sure the user doesn't already exist in the WebBoard database. Use `UserExists` to determine if a user exists.

`AddUser` returns a Boolean; true if the user is added or false if the call failed to add a user. If `AddUser` fails, it returns one of the following error codes:

- | | |
|-----|---|
| 513 | Login name missing or invalid. |
| 514 | Password missing or invalid. |
| 515 | If an ODBC error occurred when inserting the row into the database, the errors will be here. If more than one error was generated, each error will be delimited with a
 HTML tag. |

Parameters

loginName

A string containing the user's login name. It must be unique within the WebBoard database (required). Login name must be no longer than 30 characters. The first character must be one of A-Z, a-z, {}, [], ^_ (that is, the first character cannot be a number or a hyphen). All other characters must be A-Z, a-z, 0-9, {}, [], ^, \, -, _ (note

that most punctuation marks as well as spaces and tabs are prohibited in login names).

password

A string containing the user's password (required). Passwords must be no longer than 20 characters. The password can include any character except single quote, double quote, pipe, space, or tab.

email

A string containing the user's email address (optional).

firstName

A string containing the user's first name (optional).

lastName

A string containing the user's last name (optional).

city

A string containing the city part of the user's address (optional).

state

A string containing the state part of the user's address (optional).

country

A string containing the country part of the user's address (optional).

homePage

A string containing the URL of the user's home page (optional).

bio

A string containing the biographical information about the user (optional).

signature

A string containing the user's signature to show on postings (optional).

frames

Sets whether or not to use frames. Use a Boolean, where true is use frames (optional).

fullView

Sets whether to use full view or not. Use a Boolean, where true is use full view (optional).

paging

Sets whether the user receives pages. Use a Boolean, where true is accept pages (optional).

reverseTopics

Sets topic order. Use a Boolean, where true is reverse topic order (optional).

hideEmail

Sets whether or not the user's email address is hidden. Use a Boolean, where true is hide email address (optional).

autoMarkNew

Turns on and off automatic marking of read messages. Use a Boolean, where true is mark messages automatically (optional).

Failed

Failed

Causes the authentication process to fail, returning the user to the authentication failed page.

NewUser

NewUser

Stops the authentication process, returning the user to the authentication failed page, and asking if the user wants to log in as a new user or retype the login information.

Passed

Passed

Approves the authentication process, sending the user to the board's welcome page.

UserExists

UserExists(loginName, password) As Boolean

Checks the WebBoard database to see if the specified login name exists paired with the specified password.

Parameters**loginName**

A string containing the login name to check for.

password

A string containing the password to check for.

Properties**loginName()**

Returns a string containing the name as entered on the login page. This method looks for the login name in the login name in the form value and in the cookie.

password()

Returns a string containing the password as entered on the login page. This method looks for the password in the login name in the form value and in the cookie.

Board Object

The Board object makes available properties of the current board, such as the board alias, colors, name, manager, and more.

Methods

Broadcast (sender, message)

Pages all current users of all boards.

Parameters

sender

A string containing the name to show who issued a broadcast (for example, WebBoard Administrator).

message

A string containing the message to appear as a page.

Page (sender, message)

Pages all current users of the current board.

Parameters

sender

A string containing the name to show who issued the page (for example, board manager).

message

A string containing the message to appear as a page.

Properties

alias()

Returns the board's alias as a string value, for example, *admin* for the administration board. The alias is used in the URL to the board (for example, *~admin*).

authModeld()

Returns a number as a byte representing the authentication mode of the board:

- 0 = Cookie authentication
- 1 = Basic authentication
- 2 = No authentication

chatModeld()

Returns a number as a byte representing the chat mode of the board:

- 0 = Chat turned off

- 1 = Reserved
- 2 = JavaScript Chat

colorBackground()

Returns a string containing the color specified on the Change Appearance page for the right frame background color.

colorContrast()

Returns a string containing the color specified on the Change Appearance page for the contrasting table color (headers).

colorLeftALink

Returns a string containing the color specified on the Change Appearance page for the active link color in the left frame.

colorLeftFrame()

Returns a string containing the color specified on the Change Appearance page for the left frame color.

colorLeftLink()

Returns a string containing the color specified on the Change Appearance page for the link color in the left frame.

colorLeftText()

Returns a string containing the color specified on the Change Appearance page for the text color in the left frame.

colorLeftVLink()

Returns a string containing the color configured on the Change Appearance page for the visited link color in the left frame.

colorRightLink()

Returns a string containing the color specified on the Change Appearance page for the link color in the right frame.

colorRightALink

Returns a string containing the color specified on the Change Appearance page for the active link color in the right frame.

colorRightText()

Returns a string containing the color specified on the Change Appearance page for the text color in the right frame.

colorRightVLink()

Returns a string containing the color configured on the Change Appearance page for the visited link color in the right frame.

colorTable()

Returns a string containing the color specified on the Change Appearance page for the table color.

colorToolbar()

Returns a string containing the color configured on the Change Appearance page for the toolbar.

currentUserCount()

Returns an integer representing the number of current users (active within the last 30 minutes).

forumCount()

Returns an integer representing not the total number of forums in the current board, but the number of forums the user has access to in the current board.

id()

Returns a long containing the BoardID of the current board.

isClosed()

Returns a Boolean; true if the board is closed/private or false if it is not.

logoffURL()

Returns a string containing the logoff URL.

manager()

Returns a string containing the board manager's name.

name()

Returns a string containing the board's name.

newUserCount()

Returns an integer representing the number of new users today for the current board.

newUserCountAll()

Returns an integer representing the number of new users today for all boards.

showRealNames()

Returns a Boolean; true if the board is configured to show users by their real names, false if it is configured to show users by their login names.

URL()

Returns a string containing the access URL for the board, including FQDN, port (if necessary), and board alias.

usersTodayCount()

Returns an integer representing the number of users visiting the current board.

Form Object

The Form object lets your script read fields from a post and, under some conditions, set new values.

Methods

ContainsField(name)

Returns a Boolean; true if the specified field name exists in the form submitted.

Parameter**name**

A string containing the name of the field on the HTML form.

GetFormValueByName(name, *index*)

Looks up a value from the HTML form field by its name and, optionally, the index of the value for fields with multiple selections, and returns the value as a string.

Parameters**name**

A string containing the name of the field on the HTML form.

index

The index of the value for multiple-selection fields; 0 for single-selection fields. (Optional)

GetFormValueCount(name)

Looks up a multiple-selection HTML form field by name and returns a long containing the number of selected entries in the field. For example, it returns the number of selected entries in a list box that allows multiple selections.

Parameter**name**

A string containing the name of the field on the HTML form.

Request Object

The Request object provides information such as basic authentication information, host name, user agent, IP address, cookies, referer, and other details about the HTTP request.

Methods

Cookie(name)

Looks up a cookie by name and returns a string containing the value of the cookie.

Parameter**name**

A string containing the name of the cookie to retrieve the value from.

Properties

authMethod()

Returns a string containing the AUTH_TYPE CGI variable.

authPass()

Returns a string containing the REMOTE_PASSWORD CGI variable.

authRealm()

Returns a string containing the AUTH_REALM CGI variable.

authUser()

Returns a string containing the REMOTE_USER CGI variable.

cookies()

Returns a string containing all cookies sent. See also: Cookie() method.

hostName()

Returns a string containing the REMOTE_ADDR CGI variable.

ip()

Returns a string containing the IP address of the client.

logicalPath()

Returns a string containing the logical path of the URL. This is anything after the domain name and following slash. For example, with the URL *your.site.com/product.info.html?data*, it returns `product.info.html?data`.

queryString()

Returns a string containing the query string of the URL. This is anything after the question mark. For example, with the URL *your.site.com/product.info.html?data*, it returns `data`.

referer()

Returns a string containing the referer URL.

requestMethod()

Returns a string containing the request method, GET or POST.

setFormValue(fieldName, value)

Sets the specified field to the specified string value.

userAgent()

Returns a string containing the browser type (user agent) of the browser sending the request.

Response Object

The Response object allows the script to send HTML back to the browser. The response object is not available in authentication scripts.

Methods

Send(output)

Sends HTML/text to the output buffer. You can use `Send()` as many times as you like in your script; it just builds up the buffer. The result of the buffer replaces the `<WB-SCRIPT>` tag in the HTML page.

Parameter**output**

The string to send to the output buffer. Should not contain HTML header tags or HTTP headers.

SendCancelMsg(output)

Sends a full string to the output (browser), cancelling the current request. This can be used only once, unlike the Send() method. Use SendCancelMsg() only in form scripts, not in tag scripts. See also SendCancelMsgRaw().

Parameter

output

A string that contains the full HTML of the page, from <HTML> to </HTML>, *excluding* the HTTP header.

SendCancelMsgRaw(output)

Sends a full string to the output (browser), cancelling the current request. This can be used only once, unlike the Send() method. Use SendCancelMsgRaw() only in form scripts, not in tag scripts. See also SendCancelMsg().

Parameter

output

A string that contains the full HTML of the page, from <HTML> to </HTML>, *including* the HTTP header.

Properties

getResponse()

Returns a string containing the contents of the output buffer.

Server Object

The Server object provides information on server properties; for example, product name and version, the administrator's email address, and the HTTP port. It also lets you page boards, send email, and log messages to *webboard.log*.

Methods

LogMessage(message)

Adds an entry to the *webboard.log* file.

Parameter

message

A string containing the message to add to the log file.

SendMail(from, recipient, cc, bcc, subject, body)

Sends mail to any user. The recipient, cc, and bcc parameters can have multiple addresses, delimited by commas. All parameters are required.

Parameters**from**

A string containing the email address or name/address pair to go on the From: line of the message.

recipient

A string containing the email address or name/address pair to go on the To: line on the message.

cc

A string containing the email address or name/address pair to go on the Cc: line of the message.

bcc

A string containing the email address or name/address pair to go on the Bcc: line of the message.

subject

A string containing the description to go in the Subject: line of the message.

body

A string containing the body of the message.

URLDecode(URL)

Decodes a URL-encoded string and returns a string containing the normal form of the encoded string. For example, "hello%20there%20user" becomes "hello there user".

Parameter**URL**

String to unescape.

URLEncode(URL)

Applies URL encoding rules, including escape characters, to a specified string and returns the encoded string. For example, "hello there user" becomes "hello%20there%20user".

Parameter**URL**

String to escape.

Properties**adminEmail()**

Returns a string containing the email address of the WebBoard administrator as specified on the Site Management form.

GMTOffset()

Returns a string containing the GMT offset from the current time zone in 24-hour format with a + or -. The same as the WebBoard tag <WB-TIMEOFFSET>.

HTTPport()

Returns an integer representing the HTTP port of WebBoard's internal web server.

productName()

Returns a string containing the product name, plus major and minor version numbers, for example, "WebBoard/4.1."

productVersion()

Returns a string containing the WebBoard's full version number, for example, "4.10.0030."

scriptFile()

Returns a string containing the script filename (just the filename, not the entire path). For example, "WELCOME.VBS."

timeLocal()

Returns a string containing the current time in 12-hour format. This is the same as the WebBoard tag <WB-TIMELOCAL>.

timeSavings()

Returns a string containing the daylight savings time zone as configured by the computer. This is the same as the WebBoard tag <WB-TIMESAVINGS>.

timeSystem()

Returns a string containing the current system time, based on the GMT offset. This is the same as the WebBoard tag <WB-TIMESYSTEM>.

timeZone()

Returns a string containing the time zone as configured by the computer. This is the same as the WebBoard tag <WB-TIMEZONE>.

totalRequests()

Returns a long representing the total number of WebBoard requests. This is the same as the WebBoard tag <WB-TOTALREQUESTS>.

User Object

The User object makes available user properties, such as the current user's name, email, location, and all other settings on the user profile. In addition, you can change certain prop-

erties, page the user, and more. The user object is unique in that it is the only object for which you can change certain values, for example:

```
user.city = "San Diego"
```

Properties you can change are indicated by the syntax `city = newCity` instead of `city()`, which is a read-only property.

Note

The User object is not available in authentication scripts.

Methods

Page (sender, message)

Pages the current user.

Parameters**sender**

A string containing the name to appear as sender on the page (for example, WebBoard administrator).

message

A string containing the message in the page.

Properties

bio()

Returns a string containing the text in the Hobbies field of the current user's profile.

bio = newBio

Sets the Hobbies field of the current user's profile to the specified string.

canPage()

Returns a Boolean with true if the current user allows paging, or false if the user turned paging off.

city()

Returns a string containing the text in the City/Town field of the current user's profile.

city = newCity

Sets the City/Town field of the current user's profile to the specified string.

connects()

Returns a long containing the number of logins by the current user.

country()

Returns a string containing the text in the Country field of the current user's profile.

country = newCountry

Sets the Country field of the current user's profile to the specified string.

digestMode()

Returns a byte value indicating the user's mailing list format setting. Values can be: 0=Non-digest, 1=Digest, 2=Digest/ZIPped.

digestMode = newDigest

Sets the current user's mailing list format. Use a byte value where 0=Non-digest, 1=Digest, 2=Digest/ZIPped.

email()

Returns a string containing the text in the Email address field of the current user's profile.

email = newEmail

Sets the Email address field of the current user's profile to the specified string.

firstLogin()

Returns a date value containing the date and time of the first login by the current user.

firstName()

Returns a string containing the text in the First name field of the current user's profile.

firstName = newFirstName

Sets the First name field of the current user's profile to the specified string.

frames()

Returns a Boolean; true if the current user's Use frames setting is Yes and false if it is No.

fullName()

Returns a string containing the full name of the current user (the first and last name if the board is showing users by real name, or login name if the board is showing users by their login names).

fullView()

Returns a Boolean; true if the current user's Full topic view setting is Yes or false if it is No.

hideEmail()

Returns a Boolean; true if the current user's Hide email setting is Yes and false if it is No.

homePage()

Returns a string containing the text in the Home page field of the current user's profile.

isAdmin()

Returns a Boolean; true if the current user is a WebBoard administrator or false if the user is not.

isGuest()

Returns a Boolean; true if the current user logged in as a guest or false if the user did not.

isManager()

Returns a Boolean; true if the current user is a board manager or WebBoard administrator or false if the user is not.

isModerator()

Returns a Boolean; true if the current user is a conference moderator, board manager, or WebBoard administrator, or false if the user is not.

isModeratorOf(forumID)

Returns a Boolean; true if the current user is the moderator, board manager, or WebBoard administrator of the specified conference in the current board, or false if the user is not.

Parameter**forumID**

The ID of the conference. This is stored in the database in the Forums table.

lastLogin()

Returns a date value containing the date and time of the last login by the current user.

lastName()

Returns a string containing the text from the Last name field of the current user's profile.

lastName = newLastName

Sets the Last name field of the current user's profile to the specified string.

loginName()

Returns a string containing the text from the Login name field of the current user's profile.

msgsPosted()

Returns an integer representing the number of messages posted by the current user.

newAttnMessages()

Returns a long representing the number of new messages to the current user's attention.

newMessages()

Returns a long representing the number of new messages for the current user.

password()

Returns a string containing the password of the current user.

posts()

Returns a long representing the number of posts by the current user.

reverseTopics()

Returns a Boolean; true if the current user's Reverse topic order setting is Yes and false if it is No.

signature()

Returns a string containing the text from the Signature field of the current user's profile.

signature = newSignature

Sets the Signature field of the current user's profile to the specified string.

state()

Returns a string containing the text from the State/Province field of the current user's profile.

state = newState

Sets the State name field of the current user's profile to the specified string.

userID()

Returns a long containing the user ID of the current user.

userInfo(num)

Returns a string containing the value of the UserInfo field in the current user's profile.

Parameter**num**

The number of the UserInfo field (1–5).

For example:

```
response.send "UserInfo1 = " & user.userInfo(1)
```

userInfo(num) = newInfo

Sets the specified UserInfo field in the current user's profile to the string given in newInfo.

Parameter**num**

The number of the UserInfo field (1–5).

For example:

```
'Change UserInfo1
user.userInfo(1) = "Last logged in at: " & Now

'Send UserInfo1 to output stream
response.send "UserInfo1 = " & user.userInfo(1)
```

Filesystem Object

The Filesystem object provides access to basic file manipulation routines. This object is described in detail at the Microsoft Developer's site in the information on the VBScript and JScript scripting languages. When this book went to print, the URLs were as follows (but they may have changed since):

<http://msdn.microsoft.com/scripting/vbScript/vbslang/vbstoc.htm>

and:

<http://msdn.microsoft.com/scripting/JScript/jslang/jstoc.htm>

Here are some examples showing how to copy a file in various scripting languages:

(VBScript)	FileSystem.CopyFile("c:\file.txt", "c:\copy.txt")
(JavaScript)	FileSystem.CopyFile("c:\\file.txt", "c:\\copy.txt");
(PerlScript)	\$FileSystem->CopyFile("c:\\file.txt", "c:\\copy.txt");