HTTP Module

An Ember Module providing an HTTP Client implementation.

Users Guide and Reference

Version 2.0

Feb 2017

Hughes Technologies

Although best efforts have been made to ensure the completeness and accuracy of the material presented in this document, Hughes Technologies Pty Ltd does not warrant the correctness of the information provided herein.

This software product is distributed under a dual-license scheme that provides the user with a choice of licenses. This software may be used either under the terms of the GNU Public License (GPL), or the Ember OEM License. A copy of both the GPL and the OEM license is included within the software distribution. The GPL is also available from the GNU Project's web site at http://www.gnu.org. The Ember OEM License is available from the Hughes Technologies web site at http://www.Hughes.com.au.

This document was designed to be printed on a duplex (double sided) device.

Preface

Intended Audience

This document describes the Ember implementation of a simple HTTP client library. It is assumed that the reader is familiar with the HTTP protocol and the interaction with web servers.

Document Conventions

This manual has been designed to be printed on US Letter paper. While many parts of the world utilise the A4 paper size (Australia included), it is not possible to print A4 formatted documents on US Letter paper without loss of information. However, printing of US Letter formatted documents on A4 will result in a correct representation of the document with somewhat larger margins than normal.

Throughout this manual, parts of the text have been flagged with the symbol that appears in the margin opposite this paragraph. Such pieces of text are viewed as being important. The reader should ensure that paragraphs marked as important are read even if the entire manual section is only being skimmed. Important sections will include information such as tips on improving the performance of your applications, or areas where mistakes are commonly made.

Contact Information

Further information about Ember and its related software can be found on the Hughes Technologies Web site. The web site includes the latest version of Ember, documentation, support, example software, references to customer sites, and much more. Our web site can be found at

http://www.Hughes.com.au

This page left intentionally blank

Table of Contents

Introduction

The mod_http module provides an Ember interface to a basic HTTP client library. Through the use of the API bindings outlined in this manual you will be able to perform simple GET and POST operations against a web server.

API Reference

httpInitPostRequest ( )

```
int httpInitPostRequest ( host, port, url_path)
        text host
        int port
        text url_path
```

Create and populate the internal structure required to perform a POST request. A handle to the request structure is returned. To perform the actual request the handle must be passed to the httpSendRequest( ) function.

Example :

```
$handle = httpInitPostRequest ("httpbin.org",80,"/post");
```

httpInitGetRequest ( )

```
int httpInitGetRequest ( host, port, url_path)
        text host
        int port
        text url_path
```

Create and populate the internal structure required to perform a GET request. A handle to the request structure is returned. To perform the actual request the handle must be passed to the httpSendRequest( ) function.

Example :

```
$handle = httpInitPostRequest ("httpbin.org",80,"/get?test_var=HelloWorld")
```

httpAddHeader ( )

```
httpAddHeader ( handle, header_name , header_value)
        int handle
        text header_name
        text header_value
```

Add an HTTP header to the request. The header details will be included in the HTTP request when httpSendRequest( ) is called.

Example :

```
httpAddHeader($conn, "Content-type","application/x-www-form-urlencoded");
```

httpAddPostData ( )

```
httpAddPostData ( handle , variable, value )
        int handle
        text variable
        text value
```

Add a variable to a POST request. The variable name and value will be added to the request details and will be sent with the rest of the request when httpSendRequest( ) is called.

Example :

```
httpAddPostData($conn, "first_name","Bill");
httpAddPostData($conn, "flast_name","Smith");
```

httpUrlEncode ( )

```
text httpUrlEncode ( value )
        text value
```

Produce a URL encoded version of the text value provided. All values that are passed as arguments within the URL of a GET request should be URL encoded.

Example :

```
$encoded_address = httpUrlEncode("123 Brown St, Sydney");
$handle = httpInitPostRequest ("www.example.com",80,"/check_address?address=$encoded_address");
```

httpSendRequest ( )

```
int httpSendRequest ( handle )
        int handle
```

Process the request pointed to by the handle. This call will perform the TCP connection to the web server and format all the protocol parameters that have been included in the request. A failure is indicated by a return of -1 and httpGetError( ) can be used to retrieve an error message from the library.

Example :

```
if (httpSendRequest($handle) < 0)
{
    printf("HTTP Connection failed!\n");
    printf("%s\n\n", httpGetError($conn));
    exit(1);
}
```

httpGetError ( )

```
text httpGetError ( handle )
        int handle;
```

Retrieve a text string indicating why a call to httpSendRequest( ) failed.

Example :

```
if (httpSendRequest($handle) < 0)
{
        $error_msg = httpGetError($handle)
}
```

httpReadResponse ( )

```
text httpReadResponse ( handle )
        int handle
```

Read the entire response from the web server. The response is a raw response. It is not parsed or processed in any way.

Example :

```
$response = httpReadResponse($handle);
echo("$response\n");
```

httpCleanup ( )

```
httpCleanup ( handle )
        int handle
```

Close down the connection and free up any resources that have been allocated during the execution of the HTTP request.

Example :

```
$response = httpReadResponse($handle);
httpCleanup($handle);
```

Example

A Simple POST example

```
modload "mod_http";

$conn = httpInitPostRequest("httpbin.org",80,"/post");
httpAddPostData($conn, "Variable1", “Ember Rocks!”);
httpAddPostData($conn, "Variable2","www.hughes.com.au");
httpAddHeader($conn, "Content-type","application/x-www-form-urlencoded");
if (httpSendRequest($conn) < 0)
{
	printf("HTTP Connection failed!\n");
	printf("%s\n\n", httpGetError($conn));
	exit(1);
}

$response = httpReadResponse($conn);
echo("$response\n");
httpCleanup($conn);
```

A Simple GET example

```
modload "mod_http";

$var1 = httpUrlEncode("Test String");
$conn = httpInitGetRequest("httpbin.org",80,"/get?VAR1=$var1");
if (httpSendRequest($conn) < 0)
{
	printf("HTTP Connection failed!\n");
	printf("%s\n\n", httpGetError($conn));
	exit(1);
}
$response = httpReadResponse($conn);
echo("$response\n");
httpCleanup($conn);
```