PostRequest

Method Summary

Boolean	addFile(parameterName, jsFile) Add a file to the post.
Boolean	addFile(parameterName, fileName, jsFile) Add a file to the post.
Boolean	addFile(parameterName, fileName, fileLocation) Add a file to the post.
Boolean	addHeader(headerName, value) Add a header to the request.
Boolean	addParameter(name, value) Add a parameter to the post.
void	executeAsyncRequest(username, password, workstation, domain, successCallbackMethod, errorCallbackMethod) Execute the request method asynchronous using windows authentication.
void	executeAsyncRequest(username, password, successCallbackMethod, errorCallbackMethod) Execute the request method asynchronous.
void	executeAsyncRequest(successCallbackMethod, errorCallbackMethod) Execute the request method asynchronous.
Response	executeRequest() Execute the request method.
Response	executeRequest(userName, password) Execute the request method.
Response	executeRequest(userName, password, workstation, domain) Execute a request method using windows authentication.
void	setBodyContent(content) Set the body of the request.
void	setBodyContent(content, mimeType) Set the body of the request and content mime type.
void	setCharset(charset) Set the charset used when posting.

Method Details

addFile

Boolean **addFile** (parameterName, jsFile)

Add a file to the post.

Parameters

{String} parameterName
{Object} jsFile

Returns

Boolean

Sample

```
poster.addFile('myFileParamName','manual.doc','c:/temp/manual_01a.doc')
poster.addFile(null,'postXml.xml','c:/temp/postXml.xml') // sets the xml to post

var f = plugins.file.convertToJSFile('./somefile02.txt')
if (f && f.exists()) poster.addFile('myTxtFileParamName','somefile.txt', f)

f = plugins.file.convertToJSFile('./anotherfile_v2b.txt')
if (f && f.exists()) poster.addFile('myOtherTxtFileParamName', f)
```

addFile

Boolean **addFile** (parameterName, fileName, jsFile)

Add a file to the post.

Parameters

{String} parameterName
{String} fileName
{Object} jsFile

Returns

Boolean

Sample

```
poster.addFile('myFileParamName','manual.doc','c:/temp/manual_01a.doc')
poster.addFile(null,'postXml.xml','c:/temp/postXml.xml') // sets the xml to post

var f = plugins.file.convertToJSFile('./somefile02.txt')
if (f && f.exists()) poster.addFile('myTxtFileParamName','somefile.txt', f)

f = plugins.file.convertToJSFile('./anotherfile_v2b.txt')
if (f && f.exists()) poster.addFile('myOtherTxtFileParamName', f)
```

addFile

Boolean **addFile** (parameterName, fileName, fileLocation)

Add a file to the post.

Parameters

{String} parameterName
{String} fileName
{String} fileLocation

Returns

Boolean

Sample

```
poster.addFile('myFileParamName','manual.doc','c:/temp/manual_01a.doc')
poster.addFile(null,'postXml.xml','c:/temp/postXml.xml') // sets the xml to post

var f = plugins.file.convertToJSFile('./somefile02.txt')
if (f && f.exists()) poster.addFile('myTxtFileParamName','somefile.txt', f)

f = plugins.file.convertToJSFile('./anotherfile_v2b.txt')
if (f && f.exists()) poster.addFile('myOtherTxtFileParamName', f)
```

addHeader

Boolean **addHeader** (headerName, value)

Add a header to the request.

Parameters

{String} headerName
{String} value

Returns

Boolean

Sample

```
method.addHeader('Content-type','text/xml; charset=ISO-8859-1')
```

addParameter

Boolean **addParameter** (name, value)

Add a parameter to the post.

Parameters

{String} name
{String} value

Returns

Boolean

Sample

```
poster.addParameter('name','value')
poster.addParameter(null,'value') //sets the content to post
```

executeAsyncRequest

void **executeAsyncRequest** (username, password, workstation, domain, successCallbackMethod, errorCallbackMethod)

Execute the request method asynchronous using windows authentication. Success callback method will be called when response is received. Response is sent as parameter in callback. If no response is received (request errors out), the errorCallbackMethod is called with exception message as parameter.

Parameters

{String} username - the user name
{String} password - the password
{String} workstation - The workstation the authentication request is originating from.
{String} domain - The domain to authenticate within.
{Function} successCallbackMethod - callbackMethod to be called after response is received
{Function} errorCallbackMethod - callbackMethod to be called if request errors out

Returns

void

Sample

```
method.executeAsyncRequest('username','password','mycomputername','domain',globals.successCallback,globals.
errorCallback)
```

executeAsyncRequest

void **executeAsyncRequest** (username, password, successCallbackMethod, errorCallbackMethod)

Execute the request method asynchronous. Success callback method will be called when response is received. Response is sent as parameter in callback. If no response is received (request errors out), the errorCallbackMethod is called with exception message as parameter.

Parameters

{String} username - the user name
{String} password - the password
{Function} successCallbackMethod - callbackMethod to be called after response is received
{Function} errorCallbackMethod - callbackMethod to be called if request errors out

Returns

void

Sample

```
method.executeAsyncRequest(globals.successCallback,globals.errorCallback)
```

executeAsyncRequest

void **executeAsyncRequest** (successCallbackMethod, errorCallbackMethod)

Execute the request method asynchronous. Success callback method will be called when response is received. Response is sent as parameter in callback. If no response is received (request errors out), the errorCallbackMethod is called with exception message as parameter.

Parameters

{Function} successCallbackMethod - callbackMethod to be called after response is received
{Function} errorCallbackMethod - callbackMethod to be called if request errors out

Returns

void

Sample

```
method.executeAsyncRequest(globals.successCallback,globals.errorCallback)
```

executeRequest

Response **executeRequest** ()

Execute the request method.

Returns

Response

Sample

```
var response = method.executeRequest()
```

executeRequest

Response **executeRequest** (userName, password)

Execute the request method.

Parameters

{String} userName - the user name
{String} password - the password

Returns

Response

Sample

```
var response = method.executeRequest()
```

executeRequest

Response **executeRequest** (userName, password, workstation, domain)

Execute a request method using windows authentication.

Parameters

{String} userName - the user name
{String} password - the password
{String} workstation - The workstation the authentication request is originating from.
{String} domain - The domain to authenticate within.

Returns

Response

Sample

```
var response = method.executeRequest('username','password','mycomputername','domain');
```

setBodyContent

void **setBodyContent** (content)

Set the body of the request.

Parameters

{String} content

Returns

void

Sample

```
method.setBodyContent(content)
```

setBodyContent

void **setBodyContent** (content, mimeType)

Set the body of the request and content mime type.

Parameters

{String} content
{String} mimeType

Returns

void

Sample

```
method.setBodyContent(content, 'text/xml')
```

setCharset

void **setCharset** (charset)

Set the charset used when posting. If this is null or not called it will use the default charset (UTF-8).

Parameters

charset

Returns

void

Sample

```
var client = plugins.http.createNewHttpClient();
var poster = client.createPostRequest('https://twitter.com/statuses/update.json');
poster.addParameter('status',scopes.globals.textToPost);
poster.addParameter('source','Test Source');
poster.setCharset('UTF-8');
var httpCode = poster.executeRequest(scopes.globals.twitterUserName, scopes.globals.twitterPassword).
getStatusCode() // httpCode 200 is ok
```