OptionsRequest

Apr 10, 2024 01:47

Supported Clients

SmartClient WebClient NGClient

Methods Summary

Boolean	addHeader(headerName, value)	Add a header to the request.
void	executeAsyncRequest(username, password, workstation, domain, successCallbackMethod, errorCallbackMethod)	Execute the request method asynchronous.
void	executeAsyncRequest(username, password, workstation, domain, successCallbackMethod, errorCallbackMethod, callbackExtraArgs)	Execute the request method asynchronous using windows authentication.
void	executeAsyncRequest(username, password, successCallbackMethod, errorCallbackMethod)	Execute the request method asynchronous.
void	executeAsyncRequest(username, password, successCallbackMethod, errorCallbackMethod, callbackExtraArgs)	Execute the request method asynchronous using windows authentication.
void	executeAsyncRequest (successCallbackMethod, errorCallbackMethod)	Execute the request method asynchronous.
void	executeAsyncRequest (successCallbackMethod, errorCallbackMethod, callbackExtraArgs)	Execute the request method asynchronous using windows authentication.
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.
Array	getAllowedMethods(res)	Returns the supported HTTP Request operations as a String Array
void	usePreemptiveAuthentication(b)	Whatever to use preemptive authentication (sending the credentials in the header, avoiding the server request to the client - useful when uploading files, as some http servers would cancel the first request from the client, if too big, as the authentication request to the client was not yet sent)

Methods Details

addHeader(headerName, value)

```
Add a header to the request.
```

Parameters

String	headerName	;
String	value	;

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

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

```
Execute the request method asynchronous. Success callback method will be called when response is received.
Response is sent as parameter in callback.
This Response can be a response with a different status code then just 200, it could also be 500, which is
still a valid response from the server, this won't go into the error callback.
So you need to test the Reponse.getStatusCode() for that to know if everything did go OK.
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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

executeAsyncRequest(username, password, workstation, domain, successCallbackMethod, errorCallbackMethod, callbackExtraArgs)

```
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
followed by any 'callbackExtraArgs' that were given.
This Response can be a response with a different status code then just 200, it could also be 500, which is
still a valid response from the server, this won't go into the error callback.
So you need to test the Reponse.getStatusCode() for that to know if everything did go OK.
If no response is received (request errors out, network errors), the errorCallbackMethod is called with
exception message as parameter followed by any 'callbackExtraArgs' that were given.
```

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
Array	callbackExtraArgs	extra arguments that will be passed to the callback methods; can be used to identify from which request the response arrived when using the same callback method for multiple requests. Please use only simple JSON arguments (primitive types or array/objects of primitive types)

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

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.
This Response can be a response with a different status code then just 200, it could also be 500, which is
still a valid response from the server, this won't go into the error callback.
So you need to test the Reponse.getStatusCode() for that to know if everything did go OK.
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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

executeAsyncRequest(username, password, successCallbackMethod, errorCallbackMethod, callbackExtraArgs)

```
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
followed by any 'callbackExtraArgs' that were given.
This Response can be a response with a different status code then just 200, it could also be 500, which is
still a valid response from the server, this won't go into the error callback.
So you need to test the Reponse.getStatusCode() for that to know if everything did go OK.
If no response is received (request errors out, network errors), the errorCallbackMethod is called with
exception message as parameter followed by any 'callbackExtraArgs' that were given.
```

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
Array	callbackExtraArgs	extra arguments that will be passed to the callback methods; can be used to identify from which request the response arrived when using the same callback method for multiple requests. Please use only simple JSON arguments (primitive types or array/objects of primitive types)

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
method.executeAsyncRequest(globals.successCallback,globals.errorCallback, [callIDInt])
```

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.
This Response can be a response with a different status code then just 200, it could also be 500, which is
still a valid response from the server, this won't go into the error callback.
So you need to test the Reponse.getStatusCode() for that to know if everything did go OK.
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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

executeAsyncRequest(successCallbackMethod, errorCallbackMethod, callbackExtraArgs)

```
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
followed by any 'callbackExtraArgs' that were given.
This Response can be a response with a different status code then just 200, it could also be 500, which is
still a valid response from the server, this won't go into the error callback.
So you need to test the Reponse.getStatusCode() for that to know if everything did go OK.
If no response is received (request errors out, network errors), the errorCallbackMethod is called with
exception message as parameter followed by any 'callbackExtraArgs' that were given.
```

Parameters

Function	successCallbackMethod	callbackMethod to be called after response is received
Function	errorCallbackMethod	callbackMethod to be called if request errors out
Array	callbackExtraArgs	extra arguments that will be passed to the callback methods; can be used to identify from which request the response arrived when using the same callback method for multiple requests. Please use only simple JSON arguments (primitive types or array/objects of primitive types)

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
method.executeAsyncRequest(globals.successCallback,globals.errorCallback, [callIDInt])
```

executeRequest()

`Execute the request method.`

Returns

Response

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var response = method.executeRequest()

To be able to reuse the client, the response must be
closed if the content is not read via getResponseBody
 or getMediaData:

response.close()
```

executeRequest(userName, password)

`Execute the request method.`

Parameters

String	userName	the user name
String	password	the password

Returns

Response

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var response = method.executeRequest()

To be able to reuse the client, the response must be
closed if the content is not read via getResponseBody
 or getMediaData:

response.close()
```

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

getAllowedMethods(res)

```
Returns the supported HTTP Request operations as a String Array
```

Parameters

Response res The response request to get the allowed methods from.

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var supportedOperations = request.getAllowedMethods()
application.output(supportedOperations.join(','));
```

usePreemptiveAuthentication(b)

```
Whatever to use preemptive authentication (sending the credentials in the header, avoiding the server request to
the client - useful when uploading files, as some http servers would cancel the first request from the client,
if too big,
as the authentication request to the client was not yet sent)
```

Parameters

Boolean b ;

Supported Clients

SmartClient,WebClient,NGClient

Sample