

cfajaxproxy

Description

A beta release is available as Railo Extension. More info about installation can be found [here](#). Project is hosted here: <http://www.assembla.com/wiki/show/railoAjaxProxy/>. Please visit <http://groups.google.com/group/railo-beta> and get involved!

Category

Other

Implemented

Usage Syntax

```
[bind="String"]
[afc="String"]
[jsClassName="String"]
[onError="String"]
[onSuccess="String"]
[extends="Boolean"]
[methods="String"]
/>
```

[] = *Optional attribute*

Attributes

Contents

- [cfajaxproxy](#)
 - [Description](#)
 - [Category](#)
 - [Implemented](#)
 - [Usage Syntax](#)
 - [Attributes](#)
 - [Javascript Api](#)
 - [Example Usage](#)
-
- [Create a proxy object](#)
 - [Use the proxy object instance](#)

Name	Type	Required	Default	Description
bind	String	No		Bind expression that specifies a CFC method, JavaScript function, or URL to call. You cannot use this attribute with the cfc attribute.
cfc	String	No		The CFC for which to create a proxy. You must specify a dot-delimited path to the CFC (not mapping allowed). If the cfc extend enother class also the remote accessible method of the superclass will be proxied. This attribute cannot be used with the bind attribute.
jsClassName	String	No	proxy	The name to use for the JavaScript proxy class that represents the CFC.
onError	String	No		The name of a JavaScript function to invoke when a bind, specified by the bind attribute fails. The function must take two arguments: an error code and an error message. This attribute cannot be used with a cfc attribute.
onSuccess	String	No		The name of a JavaScript function to invoke when a bind, specified by the bind attribute succeeds. The function must take at least one argument that is the value returned. If Bind is cfc or Url the result is converted to a js object before being passed to the function. If parsing fails a global error handler will alert a message. This attribute cannot be used with a cfc attribute.
extends	Boolean	No	False	If true force ajaxproxy to look for remote methods in the cfc extensions chain. Any remote method found will be added to the proxy object. This attribute cannot be used with a bind attribute.
methods	String	No		Comma delimited list of methods name. If exists only the method (if remote) specified will be exposed in the proxy object.

Javascript Api

The followign method can be called on a javascript proxy object instance.

Method	Description
setAsyncMode()	As per default the remote proxy object calls will run in asynch mode.
setCallbackHandler(function)	Function called as callback. The function takes 2 arguments: the value returned and the textStatus of the ajax call.
setErrorHandler(function)	Function called as error callback. The function takes 2 arguments: the error code and the error message.
setHTTPMethod("method")	GET or POST (default is GET) Says to the proxy obejct how to parse the result of the call to the cfc method:
setReturnFormat(format)	<ul style="list-style-type: none"> • json (default) • plain (plain text) • xml - result is an xml that will be parsed into a js object.
setSyncMode()	Convert call in synch mode. Browser will wait the ajax response to proceed.
setForm('id')	Serialize the form with the provided id and append it to the query string. Specifies the JSON format in which to return ColdFusion query data. The parameter must have one of the following values:
setQueryFormat('format')	<ul style="list-style-type: none"> • row (default) - Sends the data as a JSON object with two entries: the column names and an array of row arrays. • column - Sends the data as a JSON object that represents WDDX query format. This object has three entries: the number of rows, an array with the column names, and an object where the keys are the column names and the values are arrays containing the column data.

Example Usage

Create a proxy object

```
cfc="ajaxproxy.cfc.test" jsclassname="proxyObj" onSuccess="successCallback" onError="errorCallback"/>
```

Use the proxy object instance

```
<script></span> type="text/javascript">
var myProxy = new proxyObj();
myProxy.getData();
/* call a amethod passing parameters */
var myProxy = new proxyObj();
myProxy.getData(200, 'text');
/* You can also pass parameters like a js literal object*/
var myProxy = new proxyObj();
var args = {arg:100,arg2:200};
myProxy.getData(args);
</script>
```