Common-Controls AJAX

Version 1.7 - Last changed: 19. October 2009

Common Controls

Publisher:
SCC Informationssysteme GmbH
64367 Mühltal

Tel: +49 (0) 6151 / 13 6 31 12
Internet http://www.scc-gmbh.com

Product Site:
http://www.common-controls.com

Table of contents

1 Introduction

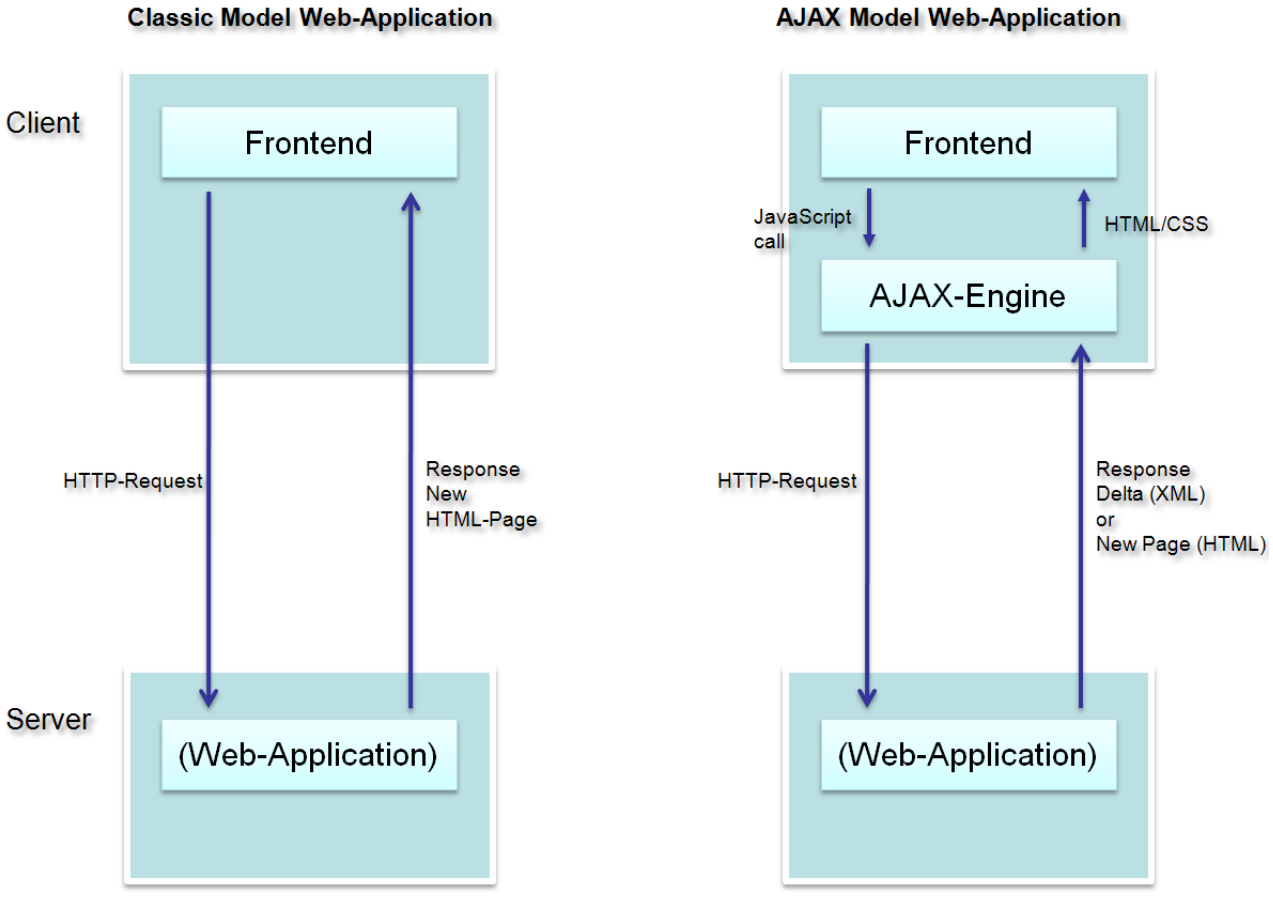
1.1 What is AJAX?

Ajax is an acronym for "**A**synchronous **J**avaScript **a**nd **X**ML". It is a concept of asynchronous data transmission between a server and the Web browser, which allows, within an HTML page, an HTTP request to carry out, without having to reload the page completely. Only parts of an HTML page are being exchanged. So the data size is much smaller than a conventional server request, which must always transmit the complete page.

Another advantage is also the fact that the displayed HTML page does not flicker. The user does not see that a server communication is carried out. This makes working with the application for the user much more pleasant.

AJAX is not a technology itself, but rather a combination of technologies such as HTML, DOM, XML and JavaScript.

The following chart shows the differences in communication with a "traditional" web application and a web application with AJAX.



Benefits:

- The current HTML page remains during an AJAX request. So the page does not "flicker". Furthermore, the html page retains its scroll position.
- The network traffic is reduced because only the modified HTML elements have to be transferred from the server to the client.
- AJAX requests can be sent synchronously or asynchronously to the server.
  Synchronously:
  The Frontend is blocked and the user can not make any entries until a server response is received.

Asynchronously:
The user can continue working with the Frontend and trigger new control element events

Disadvantages:

- JavaScript is needed. This can lead to problems with safety on the user side
- AJAX requests are not recorded in the browser history. The "back" button of the browser has therefore no useful function if it's a pure AJAX based application.
- For asynchronous AJAX requests the user can continue working with the application and send more requests to the server before a response was received. The application must therefore implement a synchronization mechanism like the Struts Token mechanism.

1.2 Operation of a Common-Controls AJAX requests

Most Controls can be configured by the "**ajax**" attribute to perform AJAX requests instead of normal HTTP requests. In this case the framework modified the generated hyperlinks that trigger the Control element actions. Each link now calls the JavaScript function **`CCAjax#intercept()`**.

This function sends the (HTTP POST) request, the control element would have normally sent to the server, as an AJAX request.

Also the HTTP parameter "**`ccAjaxId`**" (com.cc.framework.Globals.AJAX_TOKEN) will be added so that the server can distinguish the AJAX request from a normal request.

For the application developer this does not differ from the normal request handling. The framework calls just as before the corresponding event handler of the respective Struts Action.

Here again, the steps of an AJAX request in Common-Controls:

- The "ajax" attribute enables a control in the JSP page for AJAX.
- Control events are sent through the JavaScript function `CCAjax.intercept`() as an AJAX request to the server.
- The server distinguishes the AJAX request from normal requests using the "**`ccAjaxId`**" request parameter.
- A normal request processing will take place on the server.
- If the AJAX request is not cancelled by the application developer, then a XML stream with all dirty controls will be returned to the browser.
- After the AJAX call the browser will do some processing and interpret the received XML stream. Each control marked as dirty will be exchange in the current HTML page.

Process of a cancelled AJAX Request:

- The "ajax" attribute enables a control in the JSP page for AJAX.
- Control events are sent through the JavaScript function `CCAjax.intercept`() as an AJAX request to the server.
- The server distinguishes the AJAX request from normal requests using the "**`ccAjaxId`**" request parameter.
- A normal request processing will take place on the server but this time the application developer recognizes that the exchange of individual control elements is not sufficient and decides to send a new page. Therefore he can cancel the current AJAX request explicitly using the Method cancelAjaxRequest().
  Example: To edit the details of a row within a ListControl a new edit page should be displayed after the edit button gets clicked.
- The framework now quite normal forwards to the configured ActionForward (usually a JSP page).
- In the browser, the AJAX processing function is invoked. The function recognizes that there is no common controls XML structure returned from the server. So it swaps out the entire HTML DOM tree to the server response.

2 Usage within the JSP page

2.1 Configuration using the ajax attribute

The AJAX functionality of a control is configured by the "**ajax**" attribute. If it is set to true **every** control event is sent asynchronously to the server and processed there.

For complex control elements, which have nested elements in their Tag Body the AJAX feature needs to be set for the nested JSP tags too if they should carry out their events in the background.

So the AJAX feature can be set differently for the individual elements and events.

The Common-Controls Tag Library additionally provides a **<base:ajax>** tag which can be used to set the ajax property globally for all elements on the entire JSP page (see chapter 2.2). So the attribute does not need to be configured on each single element.

The following example shows the use of the AJAX property for the TreeList controls:

```
<%@ taglib uri="http://www.common-controls.com/cc/tags-ctrl"  prefix="ctrl" %>
<%@ taglib uri="http://www.common-controls.com/cc/tags-util"  prefix="util" %>

<util:imagemap name="im_region">
    <util:imagemapping  rule="country"  src="images/imgItem.gif"  width="16"  height="16"/>
</util:imagemap>

<ctrl:treelist
    name="regions"
    title="Market Hierarchy"
    rows="15"
    expandMode="multiple"
    buttons="true"
    lines="true"
    linesAtRoot="false"
    root="true"
    refreshButton="true"
    ajax="true">

    <ctrl:columntree
        title="Region"
        property="region"
        width="150"
        imageProperty="type"
        imagemap="im_region"
        ajax="true"/>

    <ctrl:columntext
        title="Name"
        property="name"
        width="300"
        sortable="true"
        ajax="true"/>

    <ctrl:columnadd     title="Add"     property="add"     permission="#admin"/>
    <ctrl:columnedit    title="Edit"    property="edit"    permission="#admin"/>

    <ctrl:columndelete  title="Delete"  property="delete"  permission="#admin" ajax="true"/>
</ctrl:treelist>
```

Figure 1: Setting the AJAX properties using the ajax attribute in the JSP page

For the columns <ctrl:columnadd/>, <ctrl:columnedit/> the "ajax" property was not set. In these cases the server should forward to a new page. The simple repaint of the control element would not be enough.

The „ajax“-attribute is responsible for:

Tag	Effect
<ctrl:treelist/>	All events that can be triggered by the buttons in the header of the control element will generate an AJAX request and cause a redrawing of the control. These include the following events: • onPage • onCreate • onPrintList • onExportList If the execution of an action should not redraw the control element, the AJAX request can be aborted. Therefore the ActionContext provides the method `cancelAjaxRequest()`. This can for example be necessary if the create button was clicked and the corresponding dialog should be presented on a new page. Example: `public void control_onCreat( ControlActionContext ctx) throws Exception { ctx.cancelAjaxRequest(); ctx.forwardByName(Forwards.CREATE); }` See also Chapter 3.3
<ctrl:columntree/>	The following events generate an AJAX request: • onExpand • onCollapse • onExpandEx • onDrilldown The column allows sending an onDrilldown-Event which should forward to a new page. If the execution of an action should not redraw the control element, the AJAX request can be aborted. Therefore the ActionContext provides the method `cancelAjaxRequest()`. Example: `public void control_onDrilldown( ControlActionContext ctx, String key) throws Exception { ctx.cancelAjaxRequest(); ctx.forwardByName(Forwards.DROLLDOWN, key); }` See also Chapter 3.3
<ctrl:columntext/>	The following events generate an AJAX request and will redraw the control: • onSort

Table 1: Impact of the AJAX-property

2.2 The <base:ajax>-Tag

The <base:ajax/> tag allows the AJAX feature to be set for:

- the entire page or
- for controls defined within the tag-body.

Furthermore the tag let you register some JavaScript functions, which should be called when receiving certain server events (response codes).

The following callback handlers are supported:

Attribute	Type	Description	Required	RTExp
onajaxerror	String	This handler will be executed when the AJAX request returned with statusCode <> 200 **Comment**: JavaScript Code		✓
onajaxsuccess	String	This handler will be executed when the AJAX request returned with statusCode = 200 **Comment**: JavaScript Code		✓
onajaxtimeout	String	If a AJAX timeout period is set, and it is reached before a response is received, a function reference assigned to this handler will be executed **Comment**: JavaScript Code		✓
timeout	Numeric String	Specifies the AJAX Timeout in milliseconds If the server does not respond within the specified time to an AJAX request, then the configured **onajaxtimeout** JavaScript handler is executed NOTE: The request-processing thread on the server is still running and will try to respond to the browser after the work is finished. But Because the browser is not waiting for the answer anymore the server will then receive a "java.net.SocketException: Connection reset" exception.		

Table 2: Attributes for the <base:ajax/>-Tag

If the <ajax/> tag is specified at the beginning of the page without the tag body, then it will set the AJAX properties for all controls on the page. So the ajax-attribute does not need to be set for each single control individually. The global settings can still be overwritten for each single element.

In the following example the onEdit and onAdd event for the <ctrl:columnedit> and <ctrl:columnadd> column will not create an AJAX request, while the other control element events are ajax-processed in the background.

```
<%@ taglib uri="http://www.common-controls.com/cc/tags-ctrl" prefix="ctrl" %>
<%@ taglib uri="http://www.common-controls.com/cc/tags-util" prefix="util" %>
<%@ taglib uri="http://www.common-controls.com/cc/tags-base" prefix="base" %>

<util:imagemap name="im region">
    <util:imagemapping  rule="country"  src="images/imgItem.gif"  width="16"  height="16"/>
</util:imagemap>

<base:ajax/>

<ctrl:treelist
    name="regions"
```

```
    title="Market Hierarchy"
    rows="15"
    expandMode="multiple"
    buttons="true"
    lines="true"
    linesAtRoot="false"
    root="true"
    refreshButton="true">

    <ctrl:columntree
        title="Region"
        property="region"
        width="150"
        imageProperty="type"
        imagemap="im region"/>

    <ctrl:columntext
        title="Name"
        property="name"
        width="300"
        sortable="true"/>

    <ctrl:columnadd     title="Add"     property="add"     permission="#admin" ajax="false"/>
    <ctrl:columnedit    title="Edit"    property="edit"    permission="#admin" ajax="false"/>
    <ctrl:columndelete  title="Delete"  property="delete"  permission="#admin"/>
</ctrl:treelist>
```

Figure 2: Setting of the AJAX property using the <base:ajax>-tag within the JSP Page

Alternatively the behaviour of individual control elements can be set by including them in the <ajax> tag body. The <ajax> tag can be nested within other <ajax> Tags.

```
<%@ taglib uri="http://www.common-controls.com/cc/tags-ctrl" prefix="ctrl" %>
<%@ taglib uri="http://www.common-controls.com/cc/tags-util" prefix="util" %>
<%@ taglib uri="http://www.common-controls.com/cc/tags-base" prefix="base" %>

<base:ajax>
    <ctrl:treelist
        name="regions"
        title="Market Hierarchy"
        rows="15"
        expandMode="multiple"
        buttons="true"
        lines="true"
        linesAtRoot="false"
        root="true"
        refreshButton="true">

        <ctrl:columntree
            title="Region"
            property="region"
            width="150"/>

        <ctrl:columntext
            title="Name"
            property="name"
            width="300"
            sortable="true"/>

        <ctrl:columnadd     title="Add"     property="add"     permission="#admin"/>
        <ctrl:columnedit    title="Edit"    property="edit"    permission="#admin"/>
        <ctrl:columndelete  title="Delete"  property="delete"  permission="#admin"/>
    </ctrl:treelist>
</base:ajax>
```

Figure 3: Setting of the AJAX property using the <base:ajax>-tag within the JSP Page

3 Action Class

3.1 ActionContext and AjaxRequest

The Common Controls Presentation Framework maps events of control elements and form elements to Java methods on the server.

The callback methods for both types derived from FWAction class will receive a special implementation of the ActionContext (see: http://www.common-controls.com/de/resources/docs/CC_Eventhandler_EN.pdf).

- If it is a Control element the method will receive a **ControlActionContext**

  Example: `control_onDrilldown(ControlActionContext ctx, String key) throws Exception`

- If it is a Form(element) the method will receive a **FormActionContext**

  Example: `buttonName_onClick(FormActionContext ctx) throws Exception`

For the server-side processing of AJAX requests a new interface **AjaxRequest** was introduced which is now a supper class of the ActionContext interface. The new methods make it possible to cancel the processing of an AJAX requests or it can be specified which controls should be redrawn. The processing on the server is no different from a normal request processing

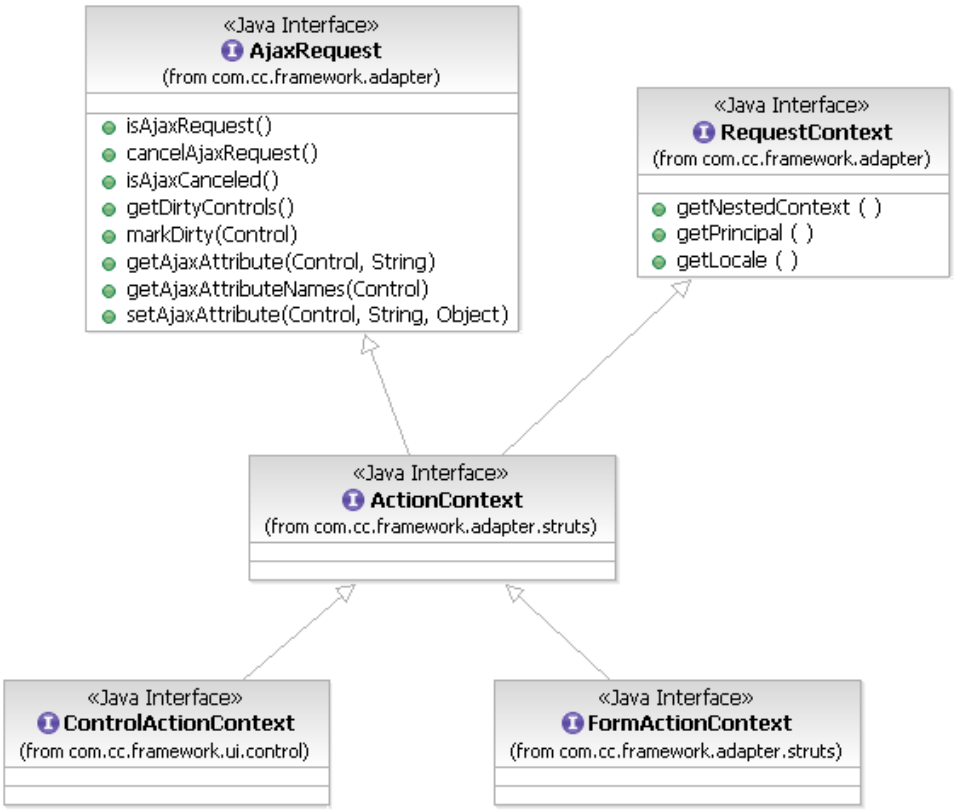


Figure 3: `Inheritance for the` **ActionContext**

The following Table shows an overview of the methods in the AjaxRequest interface. Some of the following methods will be discussed later in more detail.

Method	Description
`isAjaxRequest`	Returns true when this request was sent by an AJAX function call from within the client browser.
`cancelAjaxRequest`	Cancels the AJAX event processing of the framework. So the framework will

	not send an AJAX delta response but will forward to a normal Struts ActionForward.
`isAjaxCanceled`	Returns true when this AJAX request has been cancelled.
`getDirtyControls`	Returns an iterator that iterates all dirty controls that need to be rendered after an AJAX request.
`markDirty`	Adds the given Control to the dirty list. The context needs to be an AJAX request
`getAjaxAttribute`	Returns the attribute with the given name from the container
`getAjaxAttributeNames`	Returns an Iterator that iterates all attribute names
`setAjaxAttribute`	Adds a new Attribute to the container. Any previous existing attribute with the same name will be discarded.

Table 4: Methods from the AjaxRequest Interfaces

3.2 Control elements and Dirty-Collection

If the AJAX property of a control element within the JSP page is set, it causes a control element to redraw it self. The framework recognizes the incoming AJAX request and the control will be marked as dirty. Additionally it will be included in a list with the redraw elements.

This list can be accessed by the method `AjaxRequest#getDirtyControls()`. It is also possibility to add additional control elements to the list. For this purpose, the method `AjaxRequest#markDirty()` can be used, which expect a control element instance.

So the application developer also has the opportunity to mark dependent controls as dirty which should be updated and repainted.

The framework generates the HTML for the controls marked as dirty and sends them packed into an XML structure as a response to the AJAX request.

In the browser, the XML stream is evaluated and the corresponding elements in the DOM tree of the HTML page will be replaced (for the structure of the XML-Response see chapter 5.2).

3.3 How to cancel an AJAX-Requests

The AJAX property of a control element specifies that **all control events** should be send asynchronously to the server for processing. The server responds with an XML structure that contains the modified HTML (delta stream).

In some cases it is necessary to generate and display a complete new page. This may occur for business or technical reasons.

Examples:

1. Depending on the (technical) logic the control element should be replaced.
2. For certain events, the control element should not update itself, but forward to a new page (like the Drilldown, Edit, Add- or Create-Event).
3. After the expiry of the user session (Session Timeout), the AJAX request can not be executed anymore.
4. Upon the occurrence of an exception or after setting a message in the message collection additional content needs to be shown. In these cases a simply update for the control element is sufficient. This case is recognized and handled automatically by the Framework. Once messages in the message Collection are set, the AJAX request will be cancelled and the page will be redrawn.

If no (default) processing by the framework should take place after an AJAX-Request, the AJAX-Request must be stopped.

Therefore the ActionContext provides the method AjaxRequest#cancelAjaxRequest().

The method `AjaxRequest#isAjaxRequest()` can be used to identify an AJAX-Request.

The following example shows how to cancel an AJAX request for a drilldown event generated by the TreeList-Control. If the drilldown event occurs the navigation should forward to the detail view.

When a column gets sorted or a node is closed or expanded the control still generates an AJAX-Request to redraw it self and to prevent a flickering of the page.

a) Configuration within the JSP-Page:

```
<%@ taglib uri="http://www.common-controls.com/cc/tags-ctrl" prefix="ctrl" %>
<%@ taglib uri="http://www.common-controls.com/cc/tags-util" prefix="util" %>
<%@ taglib uri="http://www.common-controls.com/cc/tags-base" prefix="base" %>

<base:ajax/>

<ctrl:treelist
    name="regions"
    title="Market Hierarchy"
    rows="15"
    expandMode="multiple"
    buttons="true"
    lines="true"
    linesAtRoot="false"
    root="true"
    refreshButton="true">

    <ctrl:columntree
        title="Region"
        property="region"
        width="150"/>

    <ctrl:columntext
        title="Name"
        property="name"
        width="300"
        sortable="true"/>

    <ctrl:columnadd     title="Add"     property="add"     permission="#admin"/>
    <ctrl:columnedit    title="Edit"    property="edit"    permission="#admin"/>
    <ctrl:columndelete  title="Delete"  property="delete"  permission="#admin"/>
</ctrl:treelist>
```

b) Treatment of the events in the action class (excerpt):

```
public void XXXXXXXX_onDrilldown(
      ControlActionContext ctx, String key) throws Exception {

   ctx.cancelAjaxRequest();
   ctx.forwardByName(Forwards.DROLLDOWN, key);
}

public void XXXXXXXX_onCreate() {
}

public void XXXXXXXX_onEdit(

public void XXXXXXXX_onDelete(
```

3.4 Redirect [true|false]

At the end of processing a servlet can perform a **redirect** or a **forward**. This behaviour can be configured for an action forward in the struts-config.xml accordingly if the "redirect"-attribute is set to true or false.

For the execution of a redirect the method **sendRedirect()** of the Response object can be used.

- By sendRedirect the web application forces the client to **load a new URL**, which differs from the original URL.
  → Technically an HTTP 3xx code associated with the new URL is sent back to the browser, which then sends a new request addressed to that URL.
- A browser **now executes a new request** and not longer the original one.
- Since the redirect generates a second request, it is slower than a forward.
- Because it is a new request, Java Beans, which were added in the first request, are not longer accessible in the second request. But it is possible to pass Java objects the session.
- A reload (F5 button in the browser) will now execute the second, instead of the original request.
- A redirect is then useful when a new request should be generated and a reload should not go through the "original" processing.

A forward is triggered via the RequestDispatcher.

- In case of an include or forward the target servlet will get the same request as the calling servlet.
- Objects can be passed using the method request.setAttribute().
- An include or forward is performed internally by the servlet. So the web browser is not involved.
- The original URL is preserved and the web browser performs the same request again when a reload (F5 button in the browser) is executed.

When sending an AJAX request, the parameter "**ccAjaxId**" (com.cc.framework.Globals.AJAX_TOKEN) will be added the URL. The server can use this information to distinguish an AJAX request from a normal server request.

- In case of a **redirect** (redirect="true") this and all other parameters will be lost. So the AJAX-Request will be cancelled implicitly.
- In case of a **Forwards** (redirect="false") **this parameter will also be forwarded.** If the AJAX Request should be cancelled this needs to be done within the next action.

4 Implementation Examples

The online demo contains some implementation examples that illustrate the use of AJAX. Currently, there are a few examples:

- List-Control → Example 181
- Tree-Control → Example 281
- TreeList-Control → Example 381

The online demo, including the source code examples, can be downloaded from our website:
http://www.common-controls.com/de/support/update.html

5 Technical Overview

5.1 JavaScript Libraries and Functions

The Common-Controls AJAX framework provides two JavaScript libraries. The files are located in the directory `fw/jscript/ajax/`. The include statements are automatically generated by the PainterFactory.

JavaScript file	Description
ajaxrequest.js	Provides the AjaxRequest-Object, through which an AJAX request is sent.
ajax.js	Creates and processes the AJAX request. Here, the Response generated by the framework is being evaluated. As a result a new HTML page can be displayed or if a valid XML Response Protocol was received the HTML for the individual controls will be exchanged by a manipulation of the DOM.

Table 5: JavaScript files

5.2 Structure of the XML-Response Protocol

Unless a new HTML page should be displayed the following XML structure is returned as the result of the clients AJAX request. The processing takes place in the onSuccess function of CCAjax JavaScript object (see ajax.js)

```
<?xml version="1.0" encoding="UTF-8" ?>
<ajax-response>
     <token/>
     <controls>
          <control styleId="" class="" name="">
               <html>
                    <[CDATA[...]] >
               </html>
          </control>
     </controls>
</ajax-response>
```

Figure 4: Structure of the AJAX Response XML

5.3 Processing an Ajax Request

The JavaScript file **`ajax.js`** provides the necessary functions and objects to process an AJAX request. The JavaScript Object **`CCAjax`** declares the necessary callback methods, which may arise upon entering various server events.

Specifically, there exist the following functions:

Function	Description
onSuccess	This function is executed when the request executed successful. It will analyze the result returned by the server and check how the result has to be processed: • If not the expected XML-Response Protocol was received, the page will be rebuilt using the server response. • If a valid XML-Response Protocol was received the content will be analyzed and for each existing control element, the HTML will be replaced.

onError	This function is invoked in case of errors. An error message will be displayed.
onTimeOut	This function is invoked in case of a timeout. An error message will be displayed.

Table 6: Callback-Methods for an AJAX-Request