SCC
INFORMATIONSSYSTEME

COMMON CONTROLS

AJAX support

Updating control elements without a full page reload

SCC Informationssysteme GmbH

Version 1.7 · As of: 17. August 2026

Impressum

PUBLISHED BY

SCC Informationssysteme GmbH
Drosselweg 13
64367 Mühltal
+49 6151 13 63 10
www.scc-gmbh.com

Contents

1 Introduction

1.1 What is AJAX?

AJAX is an acronym for "Asynchronous JavaScript and XML". It denotes a way of transferring data asynchronously between a server and a web browser, which makes it possible to issue an HTTP request from within an HTML page without reloading the page in full. Since as a rule only parts of a page are replaced, far less data has to travel than with a conventional server request, which always has to transfer the whole page.

Another advantage is that the HTML page on display stays put during an AJAX request. The user does not see that the application is talking to the server, which makes working with it considerably more pleasant.

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

The diagram below shows how communication differs between a “classic” web application and one using AJAX.

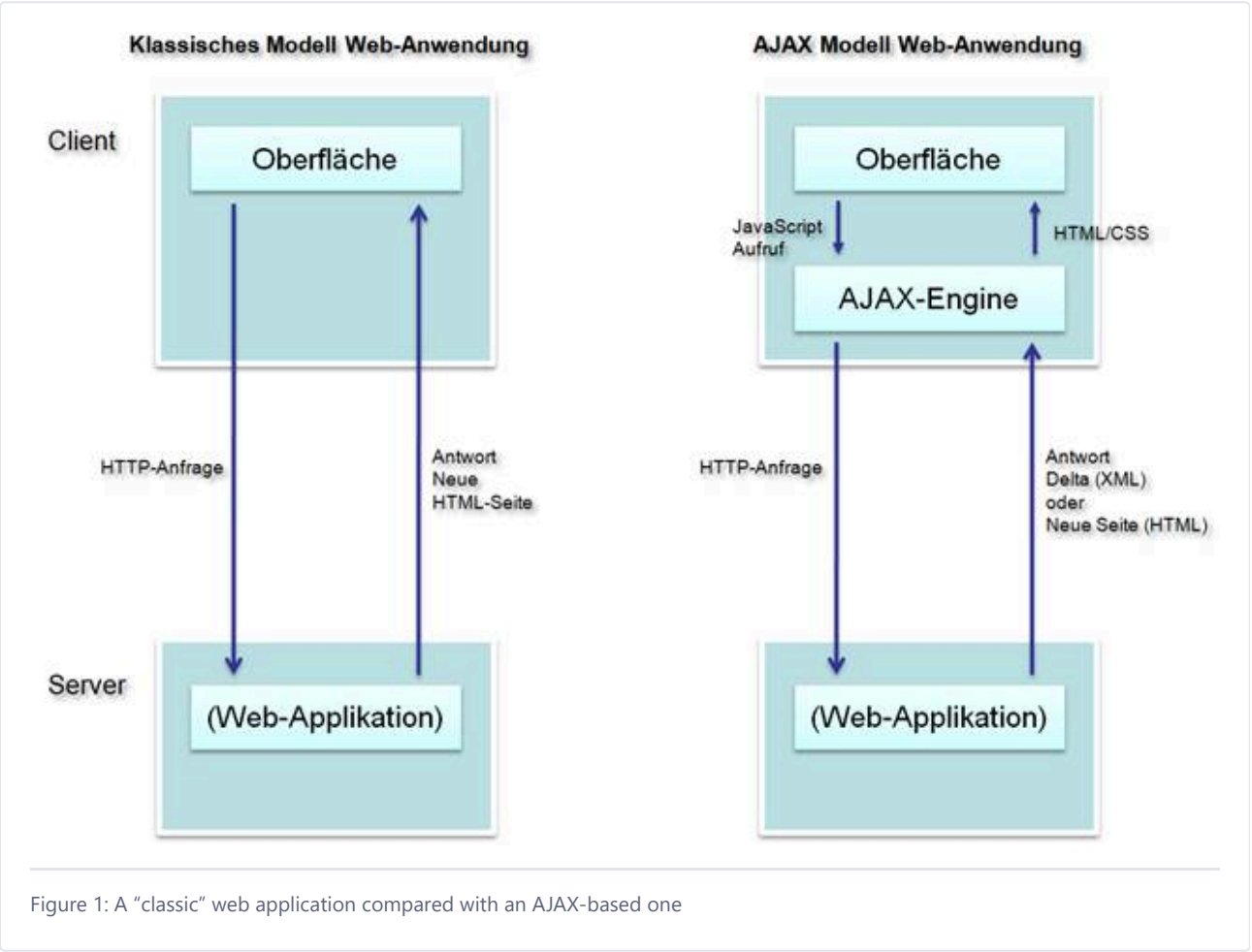

Figure 1: A “classic” web application compared with an AJAX-based one

Advantages:

· The current HTML page stays in the browser during an AJAX request, so the interface does not flicker and the page keeps its scroll position.

· Network traffic is reduced, because only the changed HTML elements have to be sent from the server.

- AJAX requests can be sent to the server synchronously or asynchronously. Synchronously: the interface blocks and the user cannot type anything until the server's answer has arrived. Asynchronously: the user can carry on working with the interface and raise further control element events, for example.

Disadvantages:

· JavaScript is indispensable, which may clash with security policies on the user's side.

· AJAX requests are not recorded in the browser history, so in a pure AJAX application the browser's back button serves no useful purpose.

- With asynchronous AJAX requests the user can carry on working and thus send further requests before the server has answered the first one. The dialogue control of the application therefore has to provide for synchronisation — by using the Struts token mechanism, for instance.

1.2 Anatomy of a Common Controls AJAX request

Most control elements can be told through the "ajax" attribute to issue AJAX requests instead of ordinary HTTP requests. The framework then modifies the generated hyperlinks that trigger the control element's actions: every hyperlink calls the JavaScript function CCAjax#intercept().

That function sends the HTTP POST request the control element would normally have sent as an AJAX request. It also passes the HTTP parameter "ccAjaxId" (com.cc.framework.Globals.AJAX_TOKEN) so that the server can tell an AJAX request from an ordinary one.

For the application developer nothing changes compared with ordinary request handling: the framework calls the event handlers of the Struts action exactly as before.

Whereas the framework normally forwards to the configured response JSP page once the action has run, in the AJAX case it sends an XML stream with the changed control elements back to the browser instead. Every control element whose state has changed enters itself into the dirty list of the request context by calling markDirty(Control). The application developer can call that method too, when further — dependent — control elements have to be redrawn.

Once more, the steps of an AJAX request in the Common Controls:

· The "ajax" attribute switches on AJAX for a control element on the JSP page.

- Events of the control element are sent to the server as AJAX requests through the JavaScript function CCAjax.intercept.
- The server tells an AJAX request from an ordinary one by the request parameter "ccAjaxId".
- The server processes the request in the usual way.
- Unless the application developer has cancelled the AJAX request, an XML stream containing all dirty control elements is sent back to the browser.
- In the browser the AJAX handling function is called; it interprets the XML stream and replaces the control elements it contains in the current HTML page.

Alternative course when the AJAX request is cancelled:

- The "ajax" attribute switches on AJAX for a control element on the JSP page.
- Events of the control element are sent to the server as AJAX requests through the JavaScript function CCAjax.intercept.
- The server tells an AJAX request from an ordinary one by the request parameter "ccAjaxId".

- The server processes the request in the usual way. This time, however, the application developer sees that replacing individual control elements is not enough and cancels the AJAX request explicitly with cancelAjaxRequest() — for example when a click on an entry of a ListControl is to lead to an editing page.
- The framework then forwards to the ActionForward set, usually a JSP page, exactly as it normally would.
- In the browser the AJAX handling function is called. It notices that no XML structure has come back from the server and therefore replaces the entire HTML DOM tree with the server's answer.

2 Use on the JSP page

2.1 Switching on through the ajax attribute

The AJAX behaviour of a control element is set through the “ajax” attribute. If it is set to true, all events of the control element are sent to the server asynchronously and processed there. For complex control elements that contain further elements in their tag body, the AJAX property has to be set on the nested JSP tags as well if their events are to run in the background too.

This makes it possible to set the AJAX property differently for individual elements and events.

So that the “ajax” attribute need not be repeated on every tag of a complex control element, the Common Controls tag library also provides the <base:ajax> tag, with which the properties can be set globally for all elements of a JSP page (see chapter 2.2).

The following example shows the AJAX property in use on a TreeListControl:

JSP

```jsp
<%@ 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>
```

```
40
```

Setting the AJAX property through the ajax attribute on the JSP page

For the columns <ctrl:columnadd/> and <ctrl:columnedit/> the “ajax” property has not been set, because in those cases the application is to move to a new page; simply redrawing the control element would not be enough.

Giving the “ajax” attribute has the following effect:

Tag	Effect

<ctrl:treelist/>	Every event that can be raised through the buttons in the header of the control element produces an AJAX request and, by default, has the control element redrawn. These are:
	onPage
	onCreate
	onPrintList
	onExportList
	If an action is not meant to redraw the control element, the AJAX request can be cancelled on the server side; the ActionContext provides the method cancelAjaxRequest() for that.
	That may be necessary when a new record is created through the create button and the corresponding dialogue appears on a new page, for example.
	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 produce an AJAX request:
	onExpand
	onCollapse
	onExpandEx
	onDrilldown
	Through a link the column can raise an onDrilldown event, which as a rule leads to a new page. Since the AJAX property of the column now makes that event merely redraw the control element instead of showing a new page, the AJAX request has to be cancelled on the server side.
	The ActionContext provides the method cancelAjaxRequest() for this. It can be called whenever not just part of the page but the whole page is to be rebuilt; calling it in the corresponding callback method cancels the AJAX request.
	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 produce an AJAX request and have the control element redrawn: onSort

Table 1: Effects of the AJAX property

2.2 The <base:ajax> tag

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

- the whole page, or
- the control elements enclosed in the tag body.

The tag can also register JavaScript functions that are called when certain server events — response codes — occur.

The following callback handlers are supported:

Attribute	Type	Description	Required	RT Exp
onajaxerror	String	This handler will be executed when the AJAX request returned with statusCode <> 200 Note: JavaScript code		✓
onajaxsuccess	String	This handler will be executed when the AJAX request returned with statusCode = 200 Note: JavaScript code		✓
onajaxtimeout	String	If an AJAX timeout period is set, and it is reached before a response is received, a function reference assigned to this handler will be executed Note: JavaScript code		✓
timeout	Numeric string	States the AJAX timeout in milliseconds. If the server does not answer an AJAX request within that time, the configured onajaxtimeout JavaScript handler is executed. Caution: the thread processing the request on the server carries on entirely unimpressed and will try to send a response back to the browser once its work is done. The server then runs into a "java.net.SocketException: Connection reset", because no browser is waiting for the answer any more.		

Table 2: Attributes of the <base:ajax/> tag

If the <ajax> tag is given without a tag body at the top of the page, it sets the AJAX properties for every control element of that page, which makes the "ajax" attribute on the individual elements unnecessary. The attribute can still be used to override the global setting for a single element: in the example below no AJAX request is produced for the onSort event of the columntext column, while all other events of the control element are handled in the background.

JSP

```jsp
<%@ 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">
```

```
21
22      <ctrl:columntree
23         title="Region"
24         property="region"
25         width="150"
26         imageProperty="type"
27         imagemap="im_region"/>
28
29      <ctrl:columntext
30         title="Name"
31         property="name"
32         width="300"
33         sortable="true"
34         ajax="false"/>
35
36      <ctrl:columnadd     title="Add"     property="add"     permission="#admin"/>
37      <ctrl:columnedit    title="Edit"    property="edit"    permission="#admin"/>
38      <ctrl:columndelete  title="Delete"  property="delete"  permission="#admin"/>
39  </ctrl:treelist>
40
```

Setting the AJAX property through the <base:ajax> tag on the JSP page

Alternatively, the behaviour of individual control elements can be governed by enclosing them in the body of an <ajax> tag; that tag may be used as often as needed on one page.

JSP

```
1  <%@ taglib uri="http://www.common-controls.com/cc/tags-ctrl" prefix="ctrl" %>
2  <%@ taglib uri="http://www.common-controls.com/cc/tags-util" prefix="util" %>
3  <%@ taglib uri="http://www.common-controls.com/cc/tags-base" prefix="base" %>
4
5  <util:imagemap name="im_region">
6      <util:imagemapping  rule="country"  src="images/imgItem.gif"  width="16"  height="16"/>
7  </util:imagemap>
8
9  <base:ajax>
10     <ctrl:treelist
11        name="regions"
12        title="Market Hierarchy"
13        rows="15"
14        expandMode="multiple"
15        buttons="true"
16        lines="true"
17        linesAtRoot="false"
18        root="true"
19        refreshButton="true">
20
21        <ctrl:columntree
22            title="Region"
23            property="region"
24            width="150"
25            imageProperty="type"
26            imagemap="im_region"/>
27
28        <ctrl:columntext
29            title="Name"
30            property="name"
31            width="300"
32            sortable="true"/>
33
34        <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>

```

Setting the AJAX property through the <base:ajax> tag on the JSP page

3 The action class

3.1 ActionContext and AjaxRequest

The Common Controls presentation framework maps events of control elements and forms onto callback methods in the action class and/or in the control element instance (see the documentation at http://www.common-controls.com/en/resources/docs/CC_Eventhandler_EN.pdf).

The signature of the callback methods contains, for an event of

- a control element, the ControlActionContext

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

- a form or form element, the FormActionContext

Example: buttonName_onClick(FormActionContext ctx) throws Exception

For handling AJAX requests on the server, the ActionContext has been extended by the interface AjaxRequest. Its methods make it possible to cancel the processing of an AJAX request or to influence which control elements are redrawn. Processing on the server is otherwise no different from ordinary request handling.

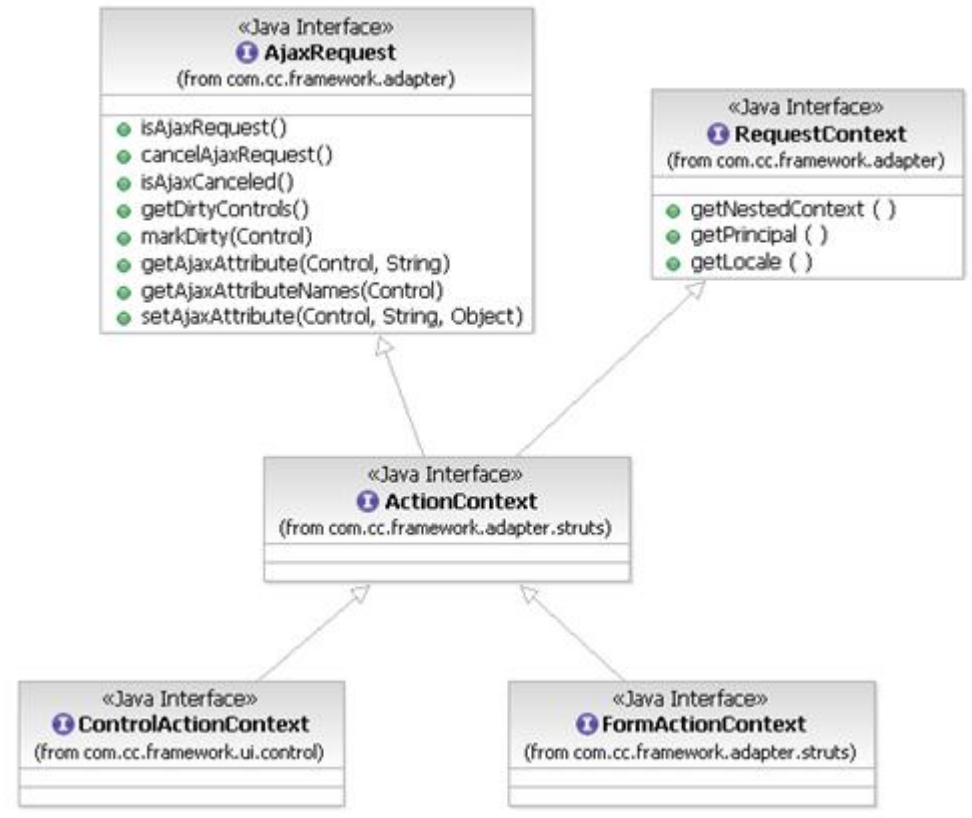

Figure 2: Inheritance hierarchy of ActionContext

The methods of the AjaxRequest interface at a glance; some of them are discussed in more detail afterwards.

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 canceled.
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 previously existing attribute with the same name will be discarded.

Table 3: Methods of the AjaxRequest interface

3.2 Control elements and the dirty collection

When the AJAX property of a control element has been set on the JSP page, any event the element raises causes it to be redrawn. The framework recognises the incoming AJAX request, marks the control element as dirty and adds it to the list of elements to be redrawn.

That list can be read with the method AjaxRequest#getDirtyControls(). Further control elements can be added to it with AjaxRequest#markDirty(), which takes a control element instance.

This lets the application developer redraw dependent control elements on the page that also have to be brought up to date after a change of state.

The framework generates the HTML of the control elements marked as dirty and sends it back, wrapped in an XML structure, as the response to the AJAX request. In the browser the XML stream is evaluated and the corresponding elements in the DOM tree of the page are replaced. The structure of the XML response is described in chapter 5.2.

3.3 Cancelling an AJAX request

The AJAX property of a control element says that all its events are sent to the server asynchronously for processing. The server answers with an XML structure containing the changed HTML — the delta stream.

In some cases, however, a completely new page has to be generated and displayed. The reasons for that may be technical or lie in the business logic.

Examples:

- Depending on the business logic, the control element itself may have to be replaced.
- On certain events — drilldown, edit, add or create — the control element is not meant to refresh itself; instead the dialogue control calls for a new page.
- Once the session has timed out, the AJAX request cannot be carried out.

- When an exception occurs, or once a message has been placed in the message collection, additional information has to be shown and refreshing the control element alone is not enough. The framework recognises and handles this case by itself: as soon as there are messages in the message collection, the AJAX request is cancelled and the page is redrawn.

If the framework is not to carry out its standard processing after an AJAX request, that request has to be cancelled.

The ActionContext provides the method AjaxRequest#cancelAjaxRequest() for this.

An AJAX request can be recognised with the method AjaxRequest#isAjaxRequest().

The following example shows an AJAX request being cancelled for a drilldown event of a TreeListControl. The drilldown event is to lead to the detail view, whereas turning to the next page of the list, sorting a column or opening and closing nodes should redraw the control element through an AJAX request, so that the page does not jump.

With its columntree column the TreeListControl is a special case: the AJAX property applies to every event of the column, but in this context the drilldown event calls for no refresh of the control element — so the programmer has to cancel the AJAX request explicitly.

a) configuration on the JSP page:

JSP

```jsp
<%@ 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) handling the various events in the action class (excerpt):

```java
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 its work a servlet can either redirect or forward. This is configured for the ActionForward in struts-config.xml through the "redirect" attribute, set to true or false.

A redirect is carried out with the method sendRedirect() of the response object.

• With sendRedirect the web application tells the client to load a new URL, different from the original one. Technically an HTTP 3xx code is sent back to the browser together with the new URL, whereupon the browser issues a fresh request to that URL.

• The browser thus no longer performs the original request but a new one.

• Because a redirect produces a second request, it is slower than a forward.

• JavaBeans created during the first request are consequently out of reach in the second one. Objects can be passed through the session instead, for example.

• On a reload (F5 in the browser) the second request is repeated rather than the original one.

• A redirect is useful when a new request is wanted and a reload should no longer run through the original processing.

A forward is initiated through the RequestDispatcher:

· With both inclusion and forwarding, the target servlet or JSP receives the same request and response objects as the calling servlet.

· Data can be passed with request.setAttribute().

· An include or a forward happens inside the servlet; the client sees nothing of it.

· The original URL is kept, and on a reload (F5 in the browser) the client repeats the very same request.

With an AJAX request, the control element also passes the parameter "ccAjaxId" (com.cc.framework.Globals.AJAX_TOKEN) to the server, so that the server can tell it from an ordinary request.

- In the case of a redirect (redirect="true") that parameter is lost along with all other request parameters — the AJAX request is thus cancelled implicitly.
- In the case of a forward (redirect="false") the parameter is passed on. If the AJAX request is to be cancelled, that has to happen in the next action.

4 Implementation examples

The online demo contains a number of examples that illustrate the use of AJAX.

At present these are:

- ListControl — example 181
- TreeControl — example 281
- TreeListControl — example 381

The online demo, source code examples included, can be downloaded from our website.

http://www.common-controls.com/en/support/update.html

5 Technical overview

5.1 JavaScript libraries and functions

For its AJAX integration the Common Controls framework provides two JavaScript libraries. The files are located in the directory fw/ajax/jscript; the include statements are added to the page automatically by the painter factory.

JavaScript file	Description
ajaxrequest.js	Provides the AjaxRequest object through which an AJAX request is sent.
ajax.js	Creates and handles the AJAX request, evaluating the response generated by the framework. That response may contain a whole new HTML page, or the HTML of individual control elements; in the latter case the parts to be redrawn are replaced by manipulating the DOM.

Table 4: JavaScript files

5.2 Structure of the XML response protocol

Unless a new HTML page is to be shown, an AJAX request returns the following XML structure to the client. It is handled in the onSuccess function of the JavaScript object CCAjax (see ajax.js).

JSP

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

```

Structure of the AJAX response XML

The CDATA section contains the HTML of the control element marked as dirty, ready to be swapped in.

5.3 Handling the AJAX request

The JavaScript file ajax.js provides the functions and objects needed to handle an AJAX request.

The JavaScript object CCAjax declares the callback methods for the various events that can occur while an AJAX request is being sent. These are:

Function	Description
onSuccess	This function runs when the request has succeeded. It analyses what the server returned and how that has to be handled. If the expected XML response protocol did not arrive, the page is rebuilt and the result is shown in the document. If the XML response protocol did arrive, it is analysed and the HTML of every control element it contains is swapped in.
onError	This function is called when something goes wrong; it shows an error message.
onTimeOut	This function is called on a timeout; it shows an error message.

Table 5: Callback methods of an AJAX request