Common-Controls AJAX

Version 1.7 - Stand: 19. Oktober 2009

Common Controls

Herausgeber:
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

Inhaltsverzeichnis

1 Einleitung

1.1 Was ist AJAX?

Ajax ist ein Akronym für "**A**synchronous **J**avaScript **a**nd **X**ML". Es bezeichnet ein Konzept der asynchronen Datenübertragung zwischen einem Server und dem Web-Browser, das es ermöglicht, innerhalb einer HTML-Seite eine HTTP-Anfrage durchzuführen, ohne die Seite komplett neu laden zu müssen. Da in der Regel nur Teile einer HTML Seite ausgetauscht werden ist die zu übertragene Datenmenge deutlich geringer als bei einem herkömmlichen Server Request, welcher immer die vollständige Seite übertragen muss.

Ein weiterer Vorteil besteht auch darin, dass die angezeigte HTML Seite während eines AJAX Request „stehen" bleibt. Der Anwender sieht also nicht, dass eine Server Kommunikation durchgeführt wird. Das macht das Arbeiten mit der Anwendung für den Anwender wesentlich angenehmer.

AJAX ist selbst keine Technologie, sondern vielmehr eine Kombination aus Technologien wie HTML, DOM, XML und JavaScript.

Das folgende Diagramm zeigt die Unterschiede in der Kommunikation bei einer „klassischen" Web-Anwendung und einer Web-Anwendung mit AJAX.

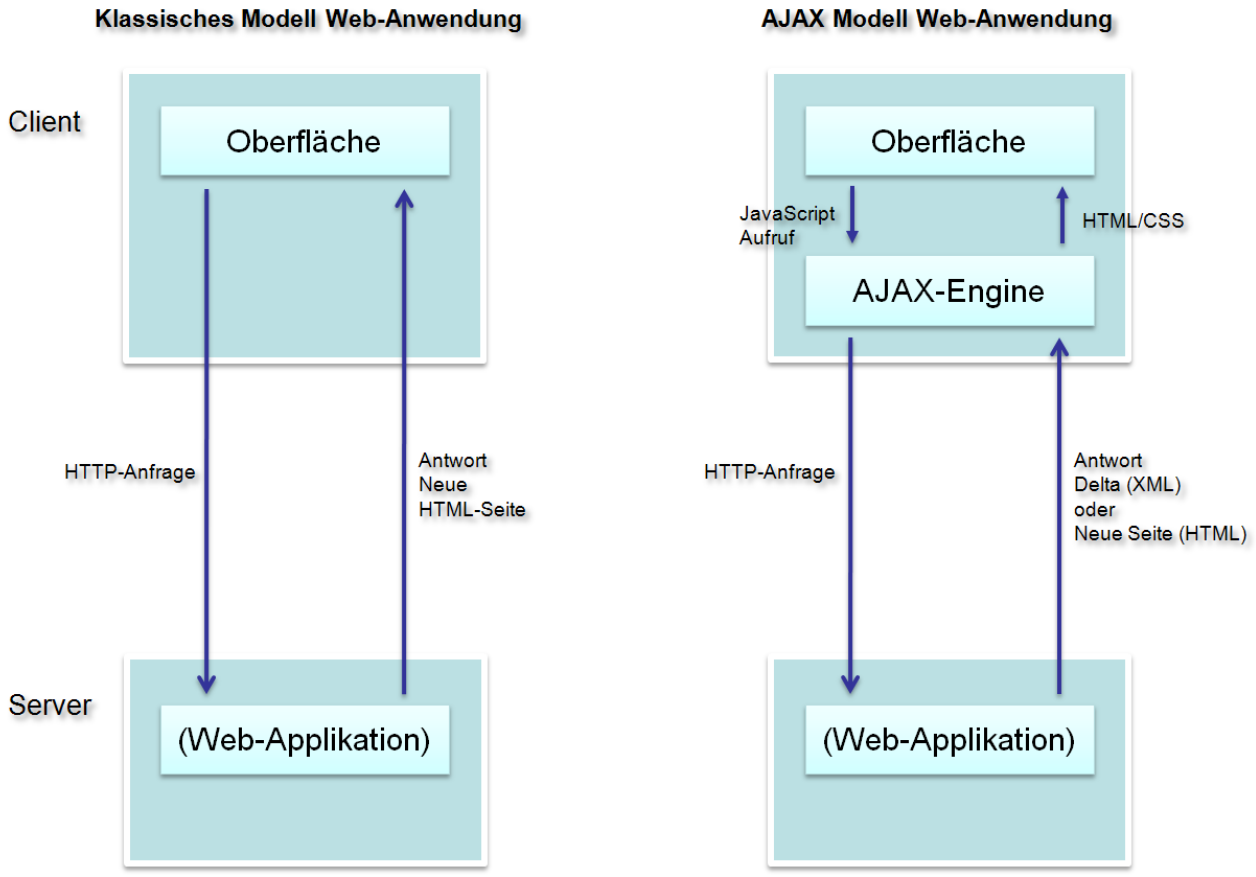

Abbildung 1: Gegenüberstellung "Klassische" und AJAX basierte Web-Anwendung

Vorteile:

- Die aktuelle HTML Seite bleibt während eines AJAX Requests im Browser stehen. Die Oberfläche „flackert" daher nicht. Außerdem behält die HTML Seite ihre Scroll Position bei.
- Der Netzwerk Verkehr wird verringert, da nur die geänderten HTML Elemente vom Server übertragen werden müssen.

- AJAX Requests können synchron oder asynchron an den Server geschickt werden.
  synchron:
  Die Oberfläche blockiert und der Anwender kann keine Eingaben machen solange bis die Server Antwort zurückgekommen ist.
  asynchron:
  Der Anwender kann weiter mit der Oberfläche arbeiten und beispielsweise weitere Kontrollelement Ereignisse auslösen.

Nachteile:

- Die Verwendung von JavaScript ist zwingend erforderlich. Das kann zu Problemen mit Sicherheitskonzepten auf der Anwenderseite führen.
- AJAX Requests werden nicht in der Browser Historie vermerkt. Die „zurück“ Schaltfläche des Browsers hat bei reinen AJAX Applikationen daher keine sinnvolle Funktion.
- Bei asynchronen AJAX Requests kann der Anwender weiterhin mit der Applikation arbeiten und so weitere Requests an den Server schicken, bevor dieser den ersten Request beantworten konnte. In der Dialogsteuerung der Anwendung müssen deshalb entsprechende Synchronisationsmechanismen vorgesehen werden (z.B. Verwendung des Struts Token Mechanismus).

1.2 Anatomie eines Common-Controls AJAX Requests

Die meisten Kontrollemente können über das „ajax“ Attribut angewiesen werden anstelle von normalen HTTP Requests AJAX Requests auszuführen. Das Framework modifiziert in diesem Fall die generierten Hyperlinks welche die Kontrollement Aktionen auslösen. Jeder Hyperlink ruft danach die JavaScript Funktion `CCAjax#intercept()` auf.

Diese Funktion schickt den (HTTP POST-) Request, den das Kontrollelement im Normalfall an den Server geschickt hätte, als AJAX Request ab. Dabei wird zusätzlich der HTTP-Parameter „`ccAjaxId`“ (com.cc.framework.Globals.AJAX_TOKEN) an den Server übergeben, damit dieser den AJAX Request von einem normalen Request unterscheiden kann.

Für den Anwendungsentwickler ergibt sich keine Änderung gegenüber einer normalen Request Behandlung. Das Framework ruft genau wie vorher die entsprechenden Event Handler der jeweiligen Struts Action auf.

Während das Framework nach der Ausführung der Action im Normalfall einfach auf die konfigurierte Antwort JSP Seite weiterleitet, wird im AJAX Fall jedoch statt dessen ein XML Stream mit den veränderten Kontrollelementen an den Browser zurückgeschickt. Jedes Kontrollelement, das seinen Zustand verändert hat hängt sich dazu in die Dirty-Liste des RequestContext ein. Dazu ruft es die Methode markDirty(Control) auf. Der Anwendungsentwickler kann diese Methode ebenfalls aufrufen, wenn weitere – abhängige - Kontrollelemente neu gezeichnet werden müssen.

Hier noch einmal die Schritte eines AJAX Requests in Common-Controls:

- Mit dem „ajax“ Attribut wird ein Kontrollelement in der JSP Seite für AJAX aktiviert.
- Kontrollelementereignisse werden über die JavaScript Funktion CCAjax.intercept als AJAX Request an den Server geschickt.
- Der Server kann den AJAX Request anhand des „ccAjaxId“ Request Parameters von normalen Requests unterscheiden.
- Es findet die normale Request Verarbeitung auf dem Server statt.
- Wenn der AJAX Request nicht vom Anwendungsentwickler abgebrochen wurde, dann wird ein XML Stream mit allen schmutzigen Kontrollelementen an den Browser zurückgeschickt.
- Im Browser wird die AJAX Bearbeitungsfunktion aufgerufen, welche den XML Stream interpretiert und die darin enthaltenen Kontrollelemente in der aktuellen HTML Seite austauscht.

Alternativer Verlauf bei abgebrochenem AJAX Request:

- Mit dem „ajax“ Attribut wird ein Kontrollelement in der JSP Seite für AJAX aktiviert.
- Kontrollelementereignisse werden über die JavaScript Funktion CCAjax.intercept als AJAX Request an den Server geschickt.
- Der Server kann den AJAX Request anhand des „ccAjaxId“ Request Parameters von normalen Requests unterscheiden.
- Es findet die normale Request Verarbeitung auf dem Server statt. Der Anwendungsentwickler erkennt diesmal aber, dass der Austausch von einzelnen Kontrollelementen nicht ausreicht. Er bricht daher den AJAX Request explizit mit cancelAjaxRequest() ab. Beispiel: Es soll nach einem Klick auf ein Element in einem ListControl auf eine Bearbeitungsseite gesprungen werden.
- Das Framework leitet nun wieder ganz normal an das gesetzte ActionForward (meist eine JSP Seite) weiter.
- Im Browser wird die AJAX Bearbeitungsfunktion aufgerufen. Diese erkennt, dass keine XML Struktur vom Server zurückgekommen ist. Sie tauscht daraufhin den gesamten HTML DOM Tree gegen die Server Antwort aus.

2 Verwendung in der JSP-Seite

2.1 Aktivierung über das ajax-Attribut

Die AJAX Funktion eines Kontrollelementes wir über das „**ajax**“-Attribut festgelegt. Wird dieses auf **true** gesetzt, dann werden **alle** Kontrollelement-Ereignisse asynchron an den Server gesendet und dort verarbeitet. Bei komplexen Kontrollelementen, die innerhalb des Tag-Bodys weitere Elemente besitzen ist die AJAX Eigenschaft entsprechend für die verschachtelten JSP-Tags zu setzen, wenn auch dort Ereignisse im Hintergrund ausgeführt werden sollen.

Dies erlaubt es die AJAX-Eigenschaft für die einzelnen Elemente und Ereignisse verschieden zu setzen.

Damit bei komplexen Kontrollelementen das „ajax“-Attribut nicht bei allen Tags angegeben werden muss, stellt die Common-Controls Tag-Bibliothek zusätzlich das **<base:ajax>-Tag** zur Verfügung, mit dem sich die Eigenschaften auch global für alle Elemente auf einer JSP Seite einstellen lassen (vgl. Kapitel 2.2).

Das folgende Beispiel zeigt die Verwendung der AJAX Eigenschaft im Falle eines 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>
```

Abbildung 2: Festlegen der AJAX-Eigenschaft über das ajax-Attribut in der JSP Seite

Für die Spalten <ctrl:columnadd/>, <ctrl:columnedit/> wurde die „ajax“-Eigenschaft nicht gesetz, da in diesen Fällen auf eine neue Seite verzweigt werden soll. Das Einfache neu zeichnen des Kontrollelementes würde hier nicht ausreichen.

Die Angabe des „ajax“-Attributes bewirkt dabei folgendes:

Tag	Auswirkung
<ctrl:treelist/>	Alle Events, die sich über die Buttons im Header des Kontrollelementes auslösen lassen, generieren einen AJAX-Request und veranlassen standardmäßig ein Neuzeichnen des Kontrollelementes. Hierzu zählen: • onPage • onCreate • onPrintList • onExportList Wenn eine Aktion kein Neuzeichnen des Kontrollelementes veranlassen soll, kann der AJAX-Request Serverseitig abgebrochen werden. Hierzu stellt der ActionContext die Methode cancelAjaxRequest() zur Verfügung. Dies kann beispielsweise bei der Anlage eines neuen Satzes über den Create-Button notwendig sein, wenn der zugehörige Dialog auf einer neuen Seite präsentiert wird. Beispiel: `public void control_onCreat( ControlActionContext ctx) throws Exception { ctx.cancelAjaxRequest(); ctx.forwardByName(Forwards.CREATE); }` Siehe auch Kapitel 3.3
<ctrl:columntree/>	Die folgenden Events generieren einen AJAX-Request: • onExpand • onCollapse • onExpandEx • onDrilldown Die Spalte ermöglicht es über einen Link ein onDrilldown-Event zu generieren, über welches in der Regel auf eine neue Seite verzweigt wird. Da dieses Event aufgrund der AJAX-Eigenschaft der Spalte nun aber lediglich ein Neuzeichnen des Kontrollelementes veranlasst, anstatt eine neue Seite zu zeigen, muss dieser (AJAX-) Request Serverseitig abgebrochen werden. Hierzu stellt der ActionContext die Methode cancelAjaxRequest() zur Verfügung. Die Methode kann immer dann aufgerufen werden, wenn nicht nur ein Teil der Seite sondern die gesamte Seite neu aufgebaut werden soll. Die Methode wird dazu in der entsprechenden Callback-Methode aufgerufen, wodurch der Abbruch des AJAX-Request veranlasst wird. Beispiel: `public void control_onDrilldown( ControlActionContext ctx, String key) throws Exception { ctx.cancelAjaxRequest(); ctx.forwardByName(Forwards.DROLLDOWN, key); }`

	Siehe auch Kapitel 3.3
<ctrl:columntext/>	Die folgenden Events generieren einen AJAX-Request und veranlassen ein Neuzeichnen des Kontrollelementes • onSort

Tabelle 1: Auswirkungen der AJAX-Eigenschaft

2.2 Das <base:ajax>-Tag

Das <base:ajax/>-Tag erlaubt es die AJAX-Eigenschaften für:

1. die gesamte Seite oder
2. für die im Tag-Body eingeschlossenen Kontrollelemente

festzulegen.

Desweiteren lassen sich mittels des Tags spezielle JavaScript-Funktionen registrieren, die beim Eintritt bestimmter Server-Events (Response-Codes) aufgerufen werden.

Folgende Callback-Handler werden unterstützt:

Attribute	Typ	Beschreibung	Pflicht	RTExp
onajaxerror	String	This handler will be executed when the AJAX request returned with statusCode <> 200 **Anmerkung**: JavaScript Code		✓
onajaxsuccess	String	This handler will be executed when the AJAX request returned with statusCode = 200 **Anmerkung**: 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 **Anmerkung**: JavaScript Code		✓
timeout	Nummerische Zeichenkette	Gibt das AJAX Timeout in Millisekunden an. Wenn der Server nicht innerhalb der angegebenen Zeit auf einen AJAX Request antwortet, dann wird der konfigurierte **onajaxtimeout** JavaScript Handler ausgeführt. ACHTUNG: Der Request verarbeitende Thread auf dem Server läuft davon aber völlig unbeeindruckt weiter und wird nach getaner Arbeit auch versuchen einen Respons an den Browser zurückzuschicken. Der Server erhält dann aber eine „java.net.SocketException: Connection reset“ Exception, weil kein Browser mehr auf die Antwort wartet.		

Tabelle 2: Attribute des <base:ajax/>-Tags

Wird das <ajax>-Tag ohne Tag Body am Seitenanfang angegeben, so legt es die AJAX Eigenschaften für alle Kontrollelemente der Seite fest. Damit wird die Angabe des „ajax“-Attributes auf Kontrollelementebene überflüssig. Das „ajax“-Attribut kann aber weiterhin dazu benutz werden, um die globale Einstellung für ein einzelnes Element zu überschreiben. Im folgenden Beispiel wird etwa bei einem onSort-Event der columntext-Spalte **kein** AJAX-Request im Hintergrund erzeugt, während die sonstigen Kontrollelement Events im Hintergrund verarbeitet werden.

```
<%@ 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"
        ajax="false"/>

    <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>
```

Abbildung 3: Festlegen der AJAX-Eigenschaft über das <base:ajax>-Tag in der JSP Seite

Alternativ kann das Verhalten einzelner Kontrollelemente beeinflusst werden, indem diese in den Tag Body eines <ajax>-Tags eingeschlossen werden. Das <ajax>-Tag lässt sich dabei beliebig oft auf einer Seite einsetzen.

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

Abbildung 4: Festlegen der AJAX-Eigenschaft über das <base:ajax>-Tag in der JSP Seite

3 Action Klasse

3.1 ActionContext und AjaxRequest

Das Common-Controls Präsentationsframeworks bildet Events von Kontrollelementen und Formularen auf Callback-Methoden innerhalb der Action-Klasse und/oder der Kontrollelementinstanz ab (Siehe hierzu Dokumentation: http://www.common-controls.com/de/resources/docs/CC_Eventhandler_DE.pdf).

Die Signatur der Callback-Methoden enthalten bei Events von einem:

- Kontrollelement den **ControlActionContext**

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

- Formular(element) den **FormActionContext**

  Beispiel: `buttonName_onClick(FormActionContext ctx) throws Exception`

Für die Serverseitige Verarbeitung von AJAX-Requests wurde der ActionContext um das Interface **AjaxRequest** erweitert. Die neuen Methoden erlauben es, die Verarbeitung eines AJAX-Requests abzubrechen oder einen Einfluss auf den Umfang der neu zu zeichnenden Kontrollelemente zu nehmen. Die Verarbeitung auf dem Server unterscheidet sich nicht von einer normalen Request Behandlung.

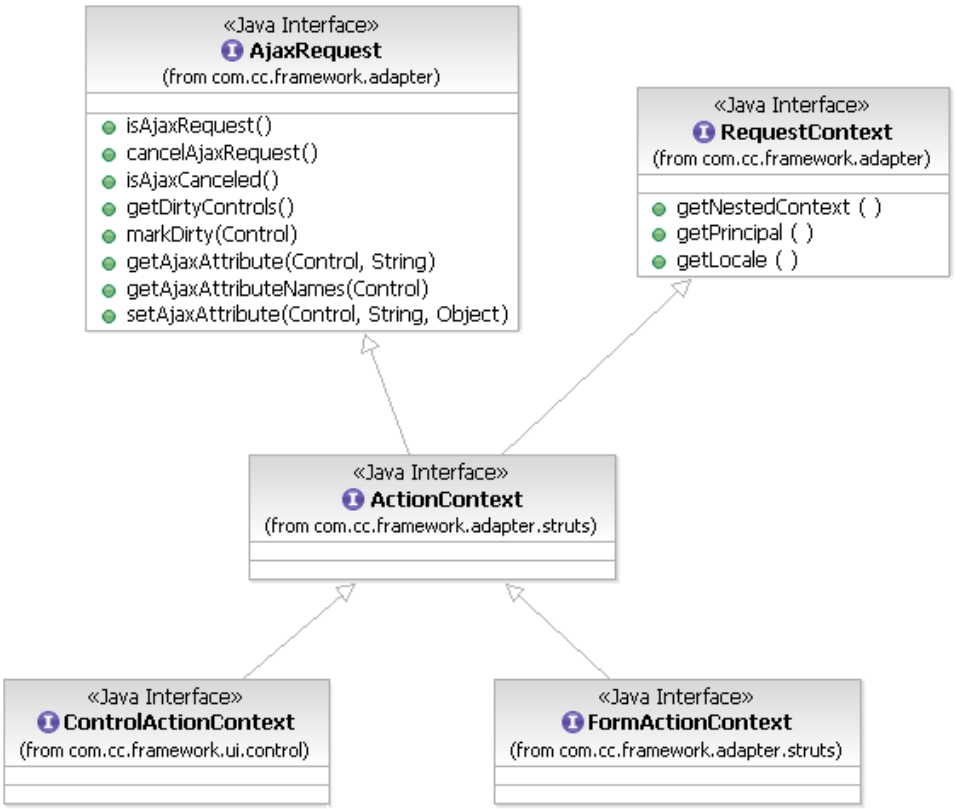

Tabelle 3: Vererbungshierarchie ActionContext

Die Methoden des AjaxRequest Interfaces im Überblick. Auf einige der Methoden wird im Anschluss genauer eingegangen.

Methode	Beschreibung
`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 needs 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.

Tabelle 4: Methoden des AjaxRequest Interfaces

3.2 Kontrollelement und Dirty-Collection

Wenn die AJAX-Eigenschaft eines Kontrollelementes innerhalb der JSP-Seite gesetzt wurde, wird bei einem durch das Kontrollelement erzeugten Event dessen Neudarstellung veranlasst. Das Framework erkennt den eingehenden AJAX-Request und markiert dazu das Kontrollelement als Dirty. Zudem wird es in eine Liste mit den neu zu zeichnenden Elementen aufgenommen.

Diese Liste kann innerhalb über die Methode `AjaxRequest#getDirtyControls()` abgefragt werden. Zudem besteht die Möglichkeit weitere Kontrollelemente der Liste hinzuzufügen. Hierfür dient die Methode `AjaxRequest#markDirty()`, welcher eine Kontrollelement-Instanz übergeben wird.

Der Anwendungsentwickler hat damit die Möglichkeit auch abhängige Kontrollelemente auf der Seite, welche durch eine Zustandsänderung ebenfalls aktualisiert werden müssen, neu zu zeichnen.

Das Framework generiert das HTML für die als dirty markierten Kontrollelemente und sendet diese, in eine XML Struktur verpackt, als Response auf den AJAX-Request zurück. Im Browser wird der XML-Stream ausgewertet und die entsprechenden Elemente im DOM Tree der HTML-Seite ersetzt (Zum Aufbau des XML-Response siehe Kapitel 5.2).

3.3 Abbrechen eines AJAX-Requests

Mit der AJAX-Eigenschaft eines Kontrollelementes wird festgelegt, dass **alle** Kontrollelement-Ereignisse asynchron zur Verarbeitung an den Server gesendet werden. Der Server antwortet mit einer XML Struktur, der das geänderte HTML (Delta-Stream) enthält.

In einigen Fällen kann es jedoch notwendig sein, dass eine komplette neue Seite generiert und angezeigt wird. Hierfür kann es fachliche oder technische Gründe geben.

Beispiele:

1. In Abhängigkeit der (fachlichen) Logik soll eventuell das Kontrollelement ausgetauscht werden
2. Bei bestimmten Events (etwa bei einem Drilldown, Edit, Add- oder Create-Event) soll das Kontrollelement sich nicht aktualisieren, sondern die Dialogsteuerung sieht vor, dass auf eine neue Seite verzweigt wird.
3. Nach dem Ablauf der Sitzungsdauer (Session TimeOut) kann der AJAX-Request nicht ausgeführt werden.
4. Bei Eintritt einer Exception oder nach dem Einstellen einer Meldung in die Message-Collection müssen zusätzliche Informationsinhalte ausgegeben werden. Ein lediglich Aktualisieren des Kontrollelementes reicht in diesen Fällen nicht aus. Dieser Fall wird von dem Framework automatisch erkannt und behandelt. *Sobald Meldungen in der Message-Collection eingestellt sind, wird der AJAX-Request abgebrochen und ein Neuzeichnen veranlasst*.

Wenn nach einem AJAX-Request keine (Standard-) Verarbeitung durch das Framework erfolgen soll, muss der AJAX-Request abgebrochen werden.

Hierzu stellt der ActionContext die Methode `AjaxRequest#cancelAjaxRequest()` bereit.

Über die Methode `AjaxRequest#isAjaxRequest()` kann ein AJAX-Request identifiziert werden.

Das folgende Beispiel zeigt den Abbruch eines AJAX-Requests im Falle eines Drilldown-Events für ein TreeList-Control. Im Falle des Drilldown-Events soll hier auf die Detail-Ansicht weiter verzweigt werden, während bei einer Navigation zur nächsten Listen-Seite, dem Sortieren einer Spalte oder dem Öffnen und Schließen von Knoten das Kontrollelement via AJAX-Request neu gezeichnet werden soll, um so ein Springen der Seite zu verhindern.

Das **TreeList-Control** nimmt hier mit der **colmntree-Spalte** eine Sonderstellung ein, da die AJAX-Eigenschaft sich hier auf alle Events der Spalte bezieht aber das Drilldown-Event in diesem Kontext keine Aktualisierung des Kontrollelementes erfordert – der Abbruch des AJAX-Requestes muss also vom Programmierer explizit vorgenommen werden.

a) Konfiguration in der JSP-Seite:

```
<%@ 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) Behandlung des verschiedener-Events in der Action-Klasse (Ausschnitt):

```
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]

Ein Servlet kann am Ende der Verarbeitung ein **Redirect** oder ein **Forward** durchführen. Dieses Verhalten lässt sich in der struts-config.xml für das ActionForward über das „**redirect**“-Attribut entsprechend mit **true** oder **false** konfigurieren.

Für die Ausführung eines Redirects dient die Methode **sendRedirect()** des Response-Objektes.

- Durch **sendRedirect** befiehlt die Web-Applikation dem Client eine **neue URL** zu laden, die sich von der ursprünglichen URL unterscheidet.
  → technisch wird ein http 3xx Code zusammen mit der neuen URL an den Browser zurückgeschickt, welcher daraufhin einen neuen Request an die genannte URL sendet.
- Ein Browser führt nun nicht mehr die ursprüngliche, sondern eine **neue Abfrage**, aus.
- Da der Redirect einen zweiten Request erzeugt ist er langsamer als ein Forward.
- Auf JavaBeans, die im ersten Request angelegt wurden, kann (folglich) im zweiten Request nicht zugegriffen werden. Objekte können beispielsweise in der Session übergeben werden.
- Bei einem Reload (F5 Taste im Browser) wird nun der zweite, anstelle des ursprünglichen Request, ausgeführt.
- Ein Redirect ist dann Nützlich, wenn ein neuer Request erzeugt werden und ein Reload nicht mehr die „ursprüngliche“ Verarbeitung durchlaufen soll.

Ein Forward wird über den RequestDispatcher angestoßen:

- Bei Einbindung (Inklusion) und Weiterleitung (Forwarding) bekommt das Ziel-Servlet/JSP den gleichen Request/Response wie das Aufrufende Servlet.
- Daten können mittels request.setAttribute() übergeben werden.
- Ein Include oder Forward wird intern vom Servlet ausgeführt. Der Client bekommt hiervon nichts mit.
- Die Ursprüngliche URL bleibt erhalten und der Client führt dieselbe Anfrage erneut aus, wenn ein Reload (F5 Taste im Browser) ausgeführt wird.

Bei einem AJAX-Request den das Kontrollelement im Normalfall an den Server schickt wird zusätzlich der Parameter „**ccAjaxId**“ (com.cc.framework.Globals.AJAX_TOKEN) an den Server übergeben, damit dieser den AJAX Request von einem normalen Request unterscheiden kann.

- Im Falle eines **Redirect** (redirect=“true“) geht der Parameter, wie alle anderen Request-Parameter, verloren – der AJAX Request wird damit also implizit abgebrochen.
- Im Falle eines **Forwards** (redirect=“false“) **wird der Parameter weitergeleitet**!
  Wenn der AJAX.-Request abgebrochen werden soll, muss dies innerhalb der nächsten Aktion erfolgen.

4 Implementierungsbeispiele

Die Online-Demo enthält einige Implementierungsbeispiele, die den Einsatz von AJAX veranschaulichen.

Derzeit existieren die folgenden Beispiele:

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

Die Online-Demo, inklusive Source Code Beispielen, kann über die WebSite heruntergeladen werden.

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

5 Technischer Überblick

5.1 JavaScript Bibliotheken und Funktionen

Das Common-Controls Framework stellt für die AJAX-Integration zwei JavaScript Bibliotheken bereit. Die Dateien befinden sich im Verzeichnis fw/ajax/jscript. Die Include Anweisungen werden über die PainterFactory automatisch in der Seite aufgenommen.

JavaScript Datei	Beschreibung
ajaxrequest.js	Stellt das AjaxRequest-Objekt zur Verfügung, über welches ein AJAX-Request versendet wird
ajax.js	Erzeugt und Verarbeitet den AJAX-Request. Hierbei wird der vom Framework generierte Respons ausgewertet. Dieser kann dabei eine neu darzustellende HTML-Seite enthalten oder es wird das HTML für einzelne Kontrollelemente übertragen. Im letzten Fall werden die neu darzustellenden Teile mittels einer Manipulation des DOM ausgetauscht.

Tabelle 5: JavaScript Dateien

5.2 Aufbau des XML ResponseProtokolls

Sofern nicht eine neue HTML-Seite dargestellt werden soll, wird als Ergebnis eines AJAX-Request die folgende XML-Struktur and den Client zurückgegeben. Die Verarbeitung erfolgt in der onSuccess Funktion des JavaScript CCAjax-Objektes (siehe ajax.js)

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

Abbildung 5: Aufbau AJAX Response XML

Im CDATA-Abschnitt ist das auszutauschende HTML des mit Dirty markierten Kontrollelementes enthalten.

5.3 Verarbeitung des AJAX-Request

Die JavaScript Datei ajax.js stellt die Notwendigen Funktionen und Objekte für die Verarbeitet eines AJAX-Request zur Verfügung.

Das JavaScript Objekt CCAjax deklariert die Callback-Methoden beim Eintreten verschiedener Ereignisse im Zusammenhang mit der Versendung eines AJAX-Request eintreten können. Konkret handelt es sich um die folgenden Funktionen:

Funktion	Beschreibung
onSuccess	Die Funktion wird bei Erfolgreicher Ausführung des Request ausgeführt. Es wird analysiert, welches Ergebnis der Server zurückgegeben hat und

	wie dieses verarbeitet werden muss. • Wurde nicht das erwartete XML-ResponseProtokoll empfangen, so wird die Seite neu aufgebaut, wobei das Ergebnis innerhalb des Dokumentes dargestellt wird. • Wurde das XML-ResponseProtokoll empfangen, so wird dieses inhaltlich analysiert und für jedes vorhandene Kontrollelement wird das HTML ausgetauscht.
onError	Die Funktion wird im Fehlerfall aufgerufen und es wird eine Fehlermeldung ausgegeben.
onTimeOut	Die Funktion wird bei einem Timeout aufgerufen und es wird eine Fehlermeldung ausgegeben.

Tabelle 6: Callback-Methoden AJAX-Request