**“WorkAround” para o problema da necessidade de dois clicks para acionar a ação de CommandButton, CommandLink**

**Solução 1.:**

**Fonte:** [**http://stackoverflow.com/questions/10094615/a4jcommandbutton-action-is-only-invoked-on-second-click**](http://stackoverflow.com/questions/10094615/a4jcommandbutton-action-is-only-invoked-on-second-click)

**Observação: testado no popup pnlGuiaRetObservacao.xhtml o qual é chamado na página guiaDetalheRetificacao.xhtml**

This problem is known as [JSF spec issue 790](http://java.net/jira/browse/JAVASERVERFACES_SPEC_PUBLIC-790). If you ajax-render some content which in turn contains a <h:form>, then its view state will get lost, which causes that the 1st action inside that form won't invoke anything. On the ajax response of the first action, the form will get new view state and thus any subsequent actions will succeed.

This is scheduled to be fixed for the upcoming JSF 2.2. But right now with JSF 2.0/2.1 you have to introduce a workaround. You first need to give the <h:form> a fixed ID which is yourFormId in the below example:

<h:panelGrid id="addToThisDepartmentPanel">

...

<h:form id="yourFormId">

...

Then you need to reference it in the render attribute as well:

<f:ajax ... render=":addToThisDepartmentPanel :yourFormId" />

The same story applies to <a4j:xxx> components.

<a4j:commandButton ... render=":addToThisDepartmentPanel :yourFormId" />

**Solução 2.:**

**Fonte:** [**http://stackoverflow.com/questions/11408130/hcommandbutton-hcommandlink-does-not-work-on-first-click-works-only-on-second**](http://stackoverflow.com/questions/11408130/hcommandbutton-hcommandlink-does-not-work-on-first-click-works-only-on-second)

**Observação: Não testada, indicado no link da Solução 1.**

You can use the following script to fix the Mojarra 2.0/2.1/2.2 bug (note: this doesn't manifest in MyFaces). This script will create the javax.faces.ViewState hidden field for forms which did not retrieve any view state after ajax update.

jsf.ajax.addOnEvent(function(data) {

if (data.status == "success") {

fixViewState(data.responseXML);

}

});

function fixViewState(responseXML) {

var viewState = getViewState(responseXML);

if (viewState) {

for (var i = 0; i < document.forms.length; i++) {

var form = document.forms[i];

if (form.method == "post") {

if (!hasViewState(form)) {

createViewState(form, viewState);

}

}

else { // PrimeFaces also adds them to GET forms!

removeViewState(form);

}

}

}

}

function getViewState(responseXML) {

var updates = responseXML.getElementsByTagName("update");

for (var i = 0; i < updates.length; i++) {

var update = updates[i];

if (update.getAttribute("id").match(/^([\w]+:)?javax\.faces\.ViewState(:[0-9]+)?$/)) {

return update.firstChild.nodeValue;

}

}

return null;

}

function hasViewState(form) {

for (var i = 0; i < form.elements.length; i++) {

if (form.elements[i].name == "javax.faces.ViewState") {

return true;

}

}

return false;

}

function createViewState(form, viewState) {

var hidden;

try {

hidden = document.createElement("<input name='javax.faces.ViewState'>"); // IE6-8.

} catch(e) {

hidden = document.createElement("input");

hidden.setAttribute("name", "javax.faces.ViewState");

}

hidden.setAttribute("type", "hidden");

hidden.setAttribute("value", viewState);

hidden.setAttribute("autocomplete", "off");

form.appendChild(hidden);

}

function removeViewState(form) {

for (var i = 0; i < form.elements.length; i++) {

var element = form.elements[i];

if (element.name == "javax.faces.ViewState") {

element.parentNode.removeChild(element);

}

}

}

Just include it as <h:outputScript name="some.js" target="head"> inside the <h:body> of the error page. If you can't guarantee that the page in question uses JSF <f:ajax>, which would trigger auto-inclusion of jsf.js, then you might want to add an additional if (typeof jsf !== 'undefined') check before jsf.ajax.addOnEvent() call, or to explicitly include it by

<h:outputScript library="javax.faces" name="jsf.js" target="head" />

Note that jsf.ajax.addOnEvent only covers standard JSF <f:ajax> and not e.g. PrimeFaces <p:ajax> or <p:commandXxx> as they use under the covers jQuery for the job. To cover PrimeFaces ajax requests as well, add the following:

$(document).ajaxComplete(function(event, xhr, options) {

if (typeof xhr.responseXML != 'undefined') { // It's undefined when plain $.ajax(), $.get(), etc is used instead of PrimeFaces ajax.

fixViewState(xhr.responseXML);

}

}

**Update** if you're using JSF utility library [OmniFaces](http://omnifaces.org/), it's good to know that the above has since 1.7 become part of OmniFaces. It's just a matter of declaring the following script in the <h:body>. See also the [showcase](http://showcase.omnifaces.org/scripts/FixViewState).

<h:body>

<h:outputScript library="omnifaces" name="fixviewstate.js" target="head" />

...

</h:body>