Nota. Para que me funcionase tuve que hacer las siguientes cosas:

1. Añadir la librería Dynamic Server Faces Components (0.2)
2. Añadir el Component Library del Dynamic Server Faces (0.2)
3. En el web.xml, mapear el servlet themes a , /faces/static/\*, además de a theme/\*

Adicionalmente, con las ajax zones

1. Utilizar la librería Sun Web UI, que tiene como xmlns, ui. Con la librería estándar, la que tiene el xmlns h, también funciona
2. Configurar el servlet de faces para que permita el rendering parcial[[1]](#footnote-2):

javax.faces.LIFECYCLE\_ID

com.sun.faces.lifecycle.PARTIAL

# Ajax Transaction

El objeto transacción se suele crear en el body de la página. Permite crear una transacción atómica Ajax. Con las Ajax Zone se puede conseguir el mismo efecto, pero la Ajax Zone aplica a un conjunto de elementos/controles. Así cuando queramos dotar el mismo comportamiento a varios controles puede resultar más adecuado una Ajax Zone.

El botón Show Virtual Forms nos permite, además de ver los virtual forms, ver las transacciones que haya definidas y que propiedades tienen.

Seleccionando un control, y usando la opción “configurar transacción Ajax que se ofrece en el menú contextual”, se puede definir el comportamiento del control dentro de cada una de las transacciones (si participa en render, sender, render-sender, o no participa).

La transacción se dispara por medio de javascript de la siguiente forma:

**DynaFaces.Tx.fire('pollTx');**

Un closure puede especificarse así:

DynaFaces.Tx.config.pollTx.closure= closure

Al seleccionar el objeto Transacción, este control tiene las siguientes propiedades:

## Execute

Los componentes (van en notación componente, no client id; esto es, se consideran todos los nodos del la hoja, desde el page1:html1:…)

En el articulo de referencia (https\_\_\_jsf-extensions.dev.java.net\_nonav\_mvn\_reference-ajax.ht.pdf en RedBooks, Java, Articulos) viene una explicación detallada.

Si se especifica “none”, no se ejecuta la fase de ejecución del JSF. Si se especifican algunos valores, se ejecutar para estos controles. Si se informa en blanco, se toman los controles indicados en la propiedad render.

## Inputs

Valores que deben enviarse al servidor. Si no se informa ninguno, todos los controles se envían. (van en notación componente, no client id; esto es, se consideran todos los nodos del la hoja, desde el page1:html1:…)

## Render

Los controles que van a ser actualizados en el cliente. (van en notación componente, no client id; esto es, se consideran todos los nodos del la hoja, desde el page1:html1:…)

Además de estos parámetros hay otros que pueden especificarse vía javascript, para personalizar el comportamiento de la transacción

## JavaScript

### Closure

Especifica el closure que se informará como tercer argumento en postReplace o replaceElement. Será una función, o una función que retorne una función javascript.

### PostReplace

The name of a globally scoped function that conforms to the following signature.

function postReplace(ajaxZone, innerHTML, [closure], [xjson]);

This function is called after the markup replacement for each component that needs to be

re-rendered with data from this ajax response. The optional argument closure is whatever

was passed as the closure option

Nótese que hay una función que proporciona el comportamiento por defecto.

\* Invoked after receiving an Ajax response and replacing elements in the DOM. Sends a poll request if continuePolling is true.

\* @param element The element we just replaced with new markup.

\* @param markup The markup with which we just replaced the element.

\* @param closure A closure that, when executed, sends the next poll request.

\*/

function customPostReplace(element, markup, closure) {

if (element == document.getElementById('form1:exchangeRateTable')) {

if (continuePolling == true) {

if (useClosure) {

//execute the closure to send the next poll request

closure();

}

else {

//send the next poll request without executing the closure

setTimeout(poll, pollDelay);

}

}

}

//invoke default behavior

markup.evalScripts();

}

### ReplaceElement

The name of a globally scoped function that conforms to the following signature.

function replaceElement(id, content, [closure], [xjson]);

This function is called for each component that needs to be re-rendered with data from this ajax response. The function is expected to take the content of the content argument and replace the DOM element referenced by the id argument.

Nótese que hay una función que realiza este trabajo por defecto. En este ejemplo para un determinado tipo de elemento hace un reemplazo “particular”, para el resto, el general.

/\*\* An array of id suffixes we will look for when invoking customReplace. \*/

var idSuffixesToReplace = [':currentPriceColumn:currentPriceText',':currentPriceColumn:currentPriceImage',':floatingProfitColumn:floatingProfitText',':floatingProfitColumn:floatingProfitImage'];

/\*\*

\* Function to perform custom replacing of elements.

\* This function is specified as the replaceElement option of fireAjaxTransaction, which is used to poll the server.

\* If the id parameter is that of the openPositionTable, extract from the src parameter nodes that represent the contents of the current price column and floating profit column, and replace the corresponding existing nodes in the page with the extracted nodes.

\* We do this so that other columns of the openPositionTable are not replaced, especially the close amount, which contains a dropdown.

\* If we were to replace the dropdown, it could interfere with a user's interacting with the dropdown.

\* If the id parameter is not that of the openPositionTable, simply call through to the default replaceElement function.

\* @param id The id of the element we are replacing as part of a call to fireAjaxTransaction.

\* @param src The new markup for the element.

\*/

function customReplace(id, src) {

if (id == 'form1:openPositionZone:openPositionTable') {

//create a temporary div

var temp = document.createElement('div');

//populate the temporary div with the src markup

temp.innerHTML = DynaFaces.trim(src);

//temp.firstChild is the src markup's root node

var revisedOpenTableNode = temp.firstChild;

//find and replace specific elements in the current price and floating profit columns

replaceNodesByIdSuffix(revisedOpenTableNode, idSuffixesToReplace);

}

else {

//just call the default replaceElement function

Element.replace(id, src);

}

}

### Jason

## Otros

### Asíncrono

### MethodName

### Inmediate

# Ajax Zone

## Execute

Especifica los controles que se van a tener en cuenta dentro del comportamiento Ajax. Sino se informa nada, todos los controles son considerados a priori. Si informamos un subconjunto de ellos, solo estos serán considerados para el comportamiento Ajax.

La manera de informarlos es con la cualificación que figura en el page navigator, empezando por form1. En el ejemplo, para el openAPositionPanelZone, si informamos form1:openAPositionPanelZone:orderTypeDropDown,form1:openAPositionPanelZone:openButton, estamos incluyendo el botón, lo que nos interesa porque queremos que sea la pulsación del botón quien dispare el ajax, y la combo DropDown. Esto significa que cuando se procese en el servidor, los valores de las otras combos no se recibirán. Si no ponemos nada en execute, entonces se reciben los valores de todas las combos.

En el servidor cuando se procese la solicitud ajax, se comprobaran los valores de los controles que lleguen, y si ha habido cambios se dispararán los eventos correspondientes. En el ejemplo que antes mencionaba, el evento asociado al botón y el asociado a la combo. Esto es, JSF aplica el mismo ciclo de vida, pero para los datos “reducidos” que llegan al servidor.

**Valor por defecto: todos los controles envían su dato al servidor.**

## Render

Especifica los controles que serán actualizados. En el ejemplo se indica form1:openPositionZone para la openAPositionPanelZone. Esto significa que este control será actualizado con el procesamiento realizado en el servidor. Si no se especifica nada solo se actualiza la zona Ajax (comprobar todas las combinaciones posibles de este valor en la referencia) exclusivamente. Si en este ejemplo quitamos el valor de la propiedad render, no se actualiza el panel con las posiciones abiertas, pero si refrescamos manualmente la página en el explorer, veremos como por arte de magia aparecen las posiciones abiertas que hayamos creado.

El rendering se realiza de la misma forma especificada por el propio proceso JSF.

**Valor por defecto: solo se refresca la zona ajax afectada.**

## JavaScript

En este apartado se pueden especificar una serie de métodos java que se ejecutaran en uno u otro punto, para configurar el comportamiento de la iteracción ajax.

### collectPostData

En este caso se envía, además de la información contenida en la propia zona Ajax (la indicada con el parámetro execute)[[2]](#footnote-3), otra información, para que sea considerada en el servidor. Obsérvese que se debe respetar una firma en el método, y que se concluye llamando la función que tiene el comportamiento “por defecto”. En el prerender podremos tomar este valor de la siguiente manera:

Map requestParameterMap = FacesContext.getCurrentInstance().getExternalContext().getRequestParameterMap();

String renderId = (String)requestParameterMap.get("form1:hiddenRenderId");

El javascript sería:

/\*\*

\* Collect the data to send with Ajax requests from Ajax zones.

\* In addition to the default collectPostData behavior, add the hiddenRenderId name/value pair.

\* @param ajaxZone The zone which is sending an Ajax request.

\* @param element The element the user has activated, causing the Ajax request to be sent.

\* @param outArray An array of name/value pairs to be populated; these will be sent with the Ajax request.

\*/

function customCollectPostData(ajaxZone, element, outArray) {

var name = 'form1:hiddenRenderId';

var value = document.getElementById(name).value;

outArray.push(name+'='+value);

DynaFacesZones.collectPostData(ajaxZone, element, outArray);

}

**Nota. Por defecto vacio, no se añade información adicional en el request.**

### eventType

El nombre del evento que dispara el ciclo Ajax. Si no se especifica nada, pues será onclick.

**Nota. Por defecto vacio, el evento que dispara el ciclo ajax es el onclick**

### getCallBackData

Una función (o una función que devuelve una referencia a una función). Esta información será la que figure en el parámetro closure en el método java informado en postReplace o en replaceElement

**Nota. Por defecto vacio, no hay closure.**

### inspectElement

Método javascript que indica que tipo de elementos dentro de la zona Ajax pueden disparar el evento. Si no se indica nada, cualquiera puede hacerlo, combos, botones, etc[[3]](#footnote-4).

**Nota. Por defecto vacio, cualquier control está en condiciones de disparar el ciclo ajax (siempre que dispare el evento especificado en eventType).**

/\*\*

\* The custom inspectElement function for openAPositionPanelZone.

\* If the supplied element is not an INPUT element, return false.

\* Otherwise, return the result of the default inspectElement function.

\* This function is necessary to prevent an Ajax request from being sent when the user interacts with any of the dropdowns in openAPositionPanelZone.

\* Note: openAPositionPanelZone contains the controls to open a position (not to be confused with openPositionZone, which shows all the currently open positions).

\* @param element The element to inspect so as to determine if it should send Ajax requests. This will be openAPositionPanelZone and its children.

\* @return False if element is not an INPUT element, or the result of the default inspectElement function otherwise.

\*/

function customInspectElementForPanelZone(element) {

return DynaFacesZones.inspectElementByNameAndAttribute(element, 'input');

}

### postReplace

Lo mismo indicado para la transacción Ajax.

### replaceElement

Lo mismo indicado para la transacción Ajax.

1. Se aprovecha el flujo de tratamiento de los servlets, pero prescindiendo del rendering. [↑](#footnote-ref-2)
2. El el ViewState figuran todos los valores de los controles comprimidos. Claro, solo se informarán los que se hayan configurado en el parámetro execute. Con esta acción estamos añadiendo un parámetro más sobre el viewstate. [↑](#footnote-ref-3)
3. Obsérvese que si el control, digamos un combo box, no dispone de un evento asociado en el lado del servidor, pues “parecerá” que nada este sucediendo, pero es solo eso, una apariencia, porque el evento si está llegando al servidor. [↑](#footnote-ref-4)