UsingTomahawkComponents

Using inputSuggestAjax with Shale and Clay

Preface

This tutorial is a sample setup for how to use special Tomahawk components like inputSuggestAjax in Clay. It is applicable to many other components, so all you need to do is adapt the same behaviour as outlined here.

First step

Note that due to an incompatible functionality in Shale validator that wraps all rendererers with a shalevalidatingrenderer, you need to disable shale validator for this to work. The issue is in jira (https://issues.apache.org/struts/browse/SHALE-442
) and planned to be fixed in a day or two as of writing.

First you need to add commons-chain to you project. Add it to ypu web-inf/lib directory, and then add the following to your web.xml file:

```
        <!-- Commons Chain Configuration Resources -->
        <context-param>
                <param-name>
                        org.apache.commons.chain.CONFIG_WEB_RESOURCE
                </param-name>
                <param-value>/WEB-INF/chain-config.xml</param-value>
        </context-param>
....

        <!-- Commons Chain Configuration Listener -->
        <listener>
                <listener-class>
                        org.apache.commons.chain.web.ChainListener
                </listener-class>
        </listener>
```

You can of course name you config file whatever you like, but the convention is chain-config.xml as shown here.

Then add to your existing chain-config.xml file or create a new one:

```
<?xml version="1.0" encoding="UTF-8"?>
<catalogs>
         ......

        <catalog name="clayCustomization">
                <chain name="suggestedItemsMethod">
                        <command
                                className="com.opstvedt.osseil.component.chain.myfaces.PropertyListenerCommand"
/>
                </chain>
                <chain name="itemLabelMethod">
                        <command
                                className="com.opstvedt.osseil.component.chain.myfaces.PropertyListenerCommand"
/>
                </chain>
        </catalog>
        ....

</catalogs>
```

The thing to note here is the name of the chain's. They must be the same as the attribute names of the component that you are planning to use. The inputSuggestAjax component is defined (in Clay) as:

```
        <component jsfid="s:inputSuggestAjax"
                componentType="org.apache.myfaces.InputSuggestAjax"
                extends="baseComponent">
                <description>
                        Provides an input textbox with "suggest" functionality,
                        using an ajax request to the server.
                </description>
                <attributes>
                        <set name="id" bindingType="VB">
                                <description>
                                        The developer-assigned ID of this component. The ID
                                        must be unique within the scope of the tag's
                                        enclosing naming container (e.g. h:form or
                                        f:subview). This value must be a static value.
                                </description>
                        </set>
                        <set name="value" bindingType="VB">
                                <description>
                                        The initial value of this component.
                                </description>
                        </set>
...
                        <set name="suggestedItemsMethod" bindingType="MB">
                                <description>
                                        Reference to the method which returns the suggested
                                        items
                                </description>
                        </set>
                        <set name="maxSuggestedItems" bindingType="VB">
                                <description>
                                        optional attribute to identify the max size of
                                        suggested Values. If specified in tableSuggestAjax,
                                        paginator functionality is used.
                                </description>
                        </set>
                        <set name="itemLabelMethod" bindingType="MB">
                                <description>
                                        Method which gets a suggested Object as an argument
                                        and returns a calculated String label. With this
                                        attribute it is possible to achieve the same
                                        mechanism as it can be found at select menues with
                                        the label/value pair.
                                </description>
                        </set>
...
                </attributes>
        </component>
```

So you must ensure that the names match, and also that the bindingType is MB in the configuration file for your component.

The next step is to create a PropertyListenerCommand. You need to figure out what parameters the component methods are (Type and number).

```
import java.util.HashMap;
import java.util.Map;

import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.el.MethodBinding;

import org.apache.commons.chain.Context;
import org.apache.log4j.Logger;
import org.apache.myfaces.custom.ajax.api.AjaxComponent;
import org.apache.shale.clay.component.chain.AbstractCommand;
import org.apache.shale.clay.component.chain.ClayContext;
import org.apache.shale.clay.config.beans.AttributeBean;
import org.apache.shale.clay.config.beans.ComponentBean;
import org.apache.shale.util.PropertyHelper;
```

```
public class PropertyListenerCommand extends AbstractCommand {

    /**
     * <p>Holds a cross-reference of attribute name to formal parameter.</p>
     */
    private static Map methodBindAttrs = new HashMap();

    private static final Logger LOGGER = Logger.getLogger(PropertyListenerCommand.class);

    /**
    * The number and type of the method that receives the Ajax request (i.e prefix and maxItems for
inputSuggestAjax etc)
    */
    static {
        methodBindAttrs.put("suggestedItemsMethod", new Class[]{String.class, Integer.class});
        methodBindAttrs.put("itemLabelMethod", new Class[]{Object.class});
    };

    /**
     * <p>
     * Looks to see if the {@link AttributeBean} on the {@link ClayContext} is one
     * of the custom Trinidad method binding event attributes. If it is, create a
     * <code>MethodBinding</code> and assign it to the component returning a
     * <code>true</code> value. Otherwise, return a <code>false</code>
     * value.
     * </p>
     *
     * @param context common chains
     * @return <code>true</code> if the chain is complete
     * @exception Exception propagated up to the top of the chain
     */
    public boolean execute(Context context) throws Exception {

        boolean isFinal = false;

        ClayContext clayContext = (ClayContext) context;
        if (clayContext == null) {
            throw new NullPointerException(getMessages().getMessage("clay.null.clayContext"));
        }
        AttributeBean attributeBean = clayContext.getAttribute();
        if (attributeBean == null) {
            throw new NullPointerException(getMessages().getMessage("clay.null.attributeBean"));
        }
        ComponentBean displayElement = clayContext.getDisplayElement();
        if (displayElement == null) {
            throw new NullPointerException(getMessages().getMessage("clay.null.componentBean"));
        }
        FacesContext facesContext = clayContext.getFacesContext();
        if (facesContext == null) {
            throw new NullPointerException(getMessages().getMessage("clay.null.facesContext"));
        }

        Class[] formalParameter = (Class[]) methodBindAttrs.get(attributeBean.getName());
        if (formalParameter != null && attributeBean.getValue() != null) {
            isFinal = true;

            UIComponent child = (UIComponent) clayContext.getChild();
            if (child == null) {
                throw new IllegalArgumentException(getMessages().getMessage("clay.null.childComponent"));
            }

            if (child instanceof AjaxComponent) {

                String expr = replaceMnemonic(clayContext);

                MethodBinding mb = facesContext.getApplication()
                        .createMethodBinding(expr, formalParameter);
                PropertyHelper propertyHelper = new PropertyHelper();

                propertyHelper.setValue(child, attributeBean.getName(), mb);
```

```
            } else {
                    LOGGER.error("Cannot bind " + attributeBean.getName() + " expression to a"
                        + " component not extending AjaxComponent: " + attributeBean.toString());
            }
        }

        return isFinal;
    }

}
```

And now you can use your inputSuggestAjax component. If you plan on using a converter with the itemLabelMethod like in the tomahawk sandbox example, you need to register a converter in the faces-config.xml file also and of course create it.

```
   <converter>
        <converter-id>inputSuggestAjaxConverter</converter-id>
        <converter-class>com.acme.test.converters.ZipCodeSuggestAjaxConverter</converter-class>
    </converter>
```