customToolbarButton (Swing SDK)

This element will cause a particular button to be present on the toolbar within Ephox EditLive! for Java Swing for Java Swing.

Configuration Element Tree Structure

<editLive>
<toolbars>
<toolbar>
<customToolbarButton>

```
<editLive>
    ...
     <toolbars>
          ...
          <toolbar>
               <customToolbarButton... />
          </toolbar>
          ...
     </toolbars>
     ...
</editLive>
```

Required Attributes

name

The name which uniquely defines this custom toolbar button.

text

The tooltip text for this custom toolbar button.

action

The action which this toolbar button performs when clicked on. This attribute has the following possible values:

- *insertHTMLAtCursor* - Insert the given HTML at the cursor.
- *insertHyperlinkAtCursor* - Insert the given hyperlink at the cursor.
- *raiseEvent* - Call a JavaScript function with the name specified in the value attribute. An event is also fired to the ELJBeanusing this XML configuration file. The following values will be assigned to the TextEvent sent with the event:
    - Action Command - *TextEvent.CUSTOM_ACTION*
    - Extra String - the string specified in the value attribute
    - Extra Object - A Java Map containing the attributes of the element selected when the specified tag is double clicked.
      The event sent to the ELJBean will also have the value TextEvent.CustomAction.RAISE_EVENT added to the extra int property of the event.
      When the event has been handled (by using the event method *setHandled(boolean b)*, where *b* = *true*), the JavaScript function defined in value will be called.
- *customPropertiesDialog* - Call a JavaScript function with the name specified in the value attribute. The current tag's properties are also passed to this function as a string. An event is also fired to the ELJBean using this XML configuration file. The following values will be assigned to the TextEventsent with the event:
    - Action Command - *TextEvent.CUSTOM_ACTION*
    - Extra String - the string specified in the value attribute
    - Extra Object - A Java Map containing the attributes of the element selected when the specified tag is double clicked.
      The event sent to the ELJBean will also have the value *TextEvent.CustomAction.RAISE_EVENT* added to the extra int property of the event.
      When the event has been handled (by using the event method *setHandled(boolean b)*, where *b* = *true*), the JavaScript function defined in value will be called.
- *PostDocument* - Post the content of the applet to a server side script.

For more information on these actions, please see the Creating Custom Menu and Toolbar Items article.

value

The value of this attribute depends on the value specified in the *action* attribute.

- *insertHTMLAtCursor* - value will be a string of HTML.
- *insertHyperlinkAtCursor* - value will be a URL.
- *raiseEvent* - value will be the name of the JavaScript function to call.
- *customPropertiesDialog* - value will be the name of the JavaScript function to call, passing the current tag's attributes as a string.

- *PostDocument*- the value attribute is used to specify several different parameters. Each parameter is delimited with the string ##ephox##. The following are the different parameters that can be specified through the value attribute:
    - *Post Field*
      The name of the field in the HTTP POST that EditLive! for Java Swing for Java Swing uses to POST its content.

      > This parameter is required.

    - *Post Acceptor URL*
      The URL for the POST acceptor that EditLive! for Java Swing for Java Swing for Java is to POST to.

      > The parameter is required.

    - *Response Processing*
      The operation that EditLive! for Java Swing for Java Swing is to perform with the HTTP response from the POST acceptor script. The parameter can have the following values:
        1. *saveToDisk* - Presents the user with a save file dialog, with which they can save the response to the local machine.
        2. *callback*- Passes the entire content of the HTTP response to a specified JavaScript callback function for processing.

        > This parameter is required.

    - *JavaScript Callback Function*
      The name of the JavaScript callback function to use for processing the response.

      > This parameter should only be used if the repsonse processing is set to *callback*.

The parameters specified through the *value* attribute string must appear in the order Post Field, Post Acceptor URL, Response Processing, and JavaScript Callback Function (if needed).

Example

The following parameters specified through the value attribue string would store the contents of EditLive! for Java Swing for Java Swing in a hidden HTML form field called *POST_field*, sending the contents via HTTP Post to *http://someserver/postacceptor.jsp*, then call back the JavaScript function called *JSFunction*.

```
value="POST_field##ephox##http://someserver/postacceptor.jsp##ephox## callback##ephox##JSFunction"
```

> When using the *insertHTMLAtCursor* action the HTML to be inserted must be URL encoded in the XML file. For example, <p>HTML to insert<p> becomes %3Cp%3EHTML%20to%20insert%3C/p%3E.

Conditional Attributes

enableintag

This attribute defines in which tags the function should be enabled. For example, when set to *td* the function will be enabled when the cursor is within a <td> tag (i.e. a table cell).

The **enableintag** attribute is required when using the *customPropertiesDialog* **action**. The **enableintag** attribute will not work with any of the other **action** attributes.

Optional Attributes

imageURL

The URL of the image to be placed on the toolbar button. The image should be of a .gif format and be a size of sixteen (16) pixels high and sixteen (16) pixels wide. This URL can be relative or absolute. Relative URLs are relative to the location of the page in which EditLive! for Java Swing is embedded.

designViewOnly

This attribute is used to indicate the action is only applicable to the design view of EditLive! for Java Swing. When set to true, the custom button (or menu item) will be disabled when the editor is switched to code view.

The default value is false.

Examples

The following example demonstrates how to define a custom toolbar button for use within EditLive! for Java Swing. The toolbar button defined in this example will insert HTML to insert at the cursor. Note that the value in the example below is URL encoded.

```
<editLive>
    ...
    <toolbars>
        ...
        <toolbar name="Example">
            <customToolbarButton
                name="customItem1"
                text="Custom Item"
                imageURL="http://www.someserver.com/image16x16.gif"
                action="insertHTMLAtCursor"
                value="%3Cp%3EHTML%20to%20insert%3C/p%3E" />
        </toolbar>
        ...
    </toolbars>
    ...
</editLive>
```

The following example demonstrates how to define a custom toolbar button which uses the *raiseEvent* action for use within EditLive! for Java Swing. The toolbar button defined in this example will call the JavaScript function called *eventRaised*.

```
<editLive>
   ...
   <toolbars>
      ...
      <toolbar name="ephox_insertmenu">
         <customToolbarButton
            name="customItem1"
            text="Raise Event"
            imageURL="http://www.someserver.com/image16x16.gif"
            action="raiseEvent"
            value="eventRaised" />
      </toolbar>
      ...
   </toolbars>
   ...
</editLive>
```

The following example demonstrates how to define a custom menu item which uses the *customPropertiesDialog* action for use within EditLive! for Java Swing. The toolbar button defined in this example will call the JavaScript function called *DisplayAttributes*. This toolbar button is only available when a <h1> tag is selected.

This operation will only work for one specified tag type. To specify which tag type the menu or toolbar item will appear for, use the **enableintag** attribute.

```
<editLive>
   ...
   <toolbars>
      ...
      <toolbar name="ephox_insertmenu">
            <customToolbarButton
                name="showAttributes"
                text="H1 Properties"
                action="customPropertiesDialog"
                value="DisplayAttributes"
                enableintag="h1"
            />
      </toolbar>
      ...
   </toolbars>
   ...
</editLive>
```

For an instance of EditLive! for Java Swing using the <**customToolbarButton**> created above, the following function would create a JavaScript dialog displaying each name-value pair of attributes for the <h1> tag.

```
function DisplayAttributes(properties)
{
   alert("VALUE-NAME pairs: " + properties);
}
```

The following example demonstrates how to define a custom toolbar button which uses the **PostDocument** action for use within EditLive! for Java Swing. The toolbar button defined in this example will POST the content in the field *editlive_field* to the script at *http://someserver/post/POSTacceptor.aspx*. Upon completion of the POST, the content of the HTTP response will be passed to the JavaScript callback function *JSFunction*.

```
<editLive>
   ...
   <toolbars>
      ...
      <toolbar name="Example">
         <customToolbarButton
            name="customItem1"
            text="POST Content"
            imageURL="http://www.someserver.com/image16x16.gif"
            action="PostDocument" value="editlive_field##ephox##http://someserver/post/POSTacceptor.aspx
##ephox##callback##ephox##JSFunction" />
      </toolbar>
      ...
   </toolbars>
   ...
</editLive>
```

Remarks

The <**customToolbarButton**> element can appear multiple times within the <toolbar> element.

The <**customToolbarButton**> element must be a complete tag; it cannot contain a tag body. Therefore, the tag must be closed in the same line. See the example below:

```
<customToolbarButton name=... />
```

Text assigned to the value attribute must be URL encoded as it is in the example above.

See Also

- Creating Custom Menu and Toolbar Items