Jelly form controls

WARNING: much of this document is old and suggests code patterns which are *not recommended*. As of this writing, the official documentation for this area of Jenkins has not been written. The ui-samples plugin is somewhat better maintained than this.

Jelly form controls

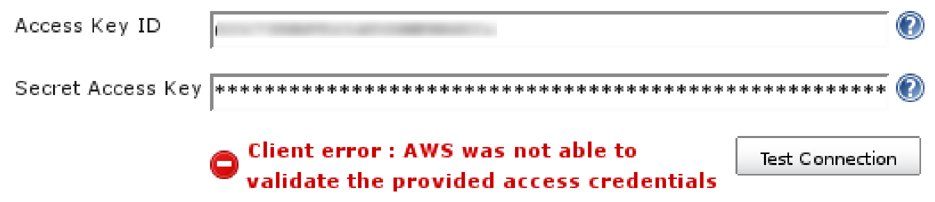
Building Tag Library Documentation

Most of the the jelly files in the Jenkins source have embedded documentation. Unfortunately, this documentation is not currently published on the web (only an old version is available). You can generate your own easily! First, get a Jenkins build going, following these directions. Then run "mvn site" which will generate the jelly tag library documentation in hudson/main/core/target/site/jelly-taglib-ref.html (on disk). If you are trying to understand Jenkins' jelly-isms, this document will help a lot.

Validation Button

This tag creates a right-aligned button for performing server-side validation. It is suitable for situations where the validation depends on the values of multiple input fields, such as credential check that uses both username and password.

Jelly code	Example

```
<f:entry title="${%Access Key ID}" help="...">
  <f:textbox field="accessId" />
</f:entry>
<f:entry title="${%Secret Access Key}" help="...">
  <f:password field="secretKey" />
</f:entry>
<f:validateButton
   title="${%Test Connection}" progress="${%Testing...}"
   method="testConnection" with="secretKey,accessId" />
```
	 Client error : AWS was not able to validate the provided access credentials

The title attribute is used to determine the text written on the button. The progress attribute determines the message displayed while the server-side validation is in progress. The method attribute specifies the server-side method invoked by this button; this follows the stapler name mangling convention, so for example `method="testConnection"` will invoke the `doTestConnection` method. This method needs to be on the `Descriptor` class that owns this form fragment.

Finally, the with attribute specifies the input fields sent to the server for the validation. They are matched against the `field` attribute or the name attribute of other input controls. The values of the nearest input fields **above** the `<f:validateButton/>` are sent to the server, so this means the button has to come after the input fields. Multiple fields can be specified by using ','.

On the server side, this tag invokes the standard "do"-style method like this:

```
public FormValidation doTestConnection(@QueryParameter("accessId") final String accessId,
        @QueryParameter("secretKey") final String secretKey) throws IOException, ServletException {
    try {
        ... do some tests ...
        return FormValidation.ok("Success");
    } catch (EC2Exception e) {
        return FormValidation.error("Client error : "+e.getMessage());
    }
}
```

The `doTestConnection` method contains the validation logic. In the end, this method has to call either `FormValidation.ok`, `.warning`, or `.error` method. Depending on which method you use, the appropriate HTML will be rendered on the client side to indicate the status to the user.

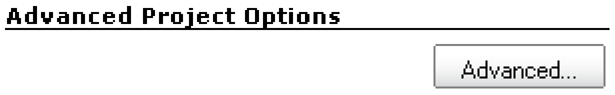
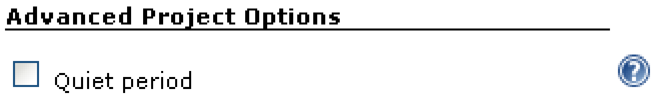
Advanced

Expandable section that shows "advanced..." button by default. Upon clicking it, a section unfolds.

Optional attributes:

- help (optional) If present, URL of the help HTML page.
- title (optional) Overrides the button's text content. If not present "Advanced" is used.

Jelly code	Example

`<f:section title="Advanced Project Options">`
`  <f:advanced>`
`    <p:config-quietPeriod />`
`    <st:include page="configure-advanced.jelly" optional="true" />`
`  </f:advanced>`
`</f:section>`
	At start only button is shown:
	
	When the button is clicked it disappears, and the content is displayed instead
	

OptionalBlock

Foldable block expanded when the menu item is checked.

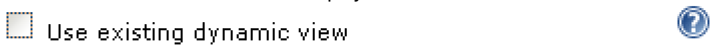
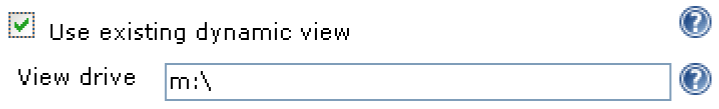
Mandatory attributes:

- name Name of the checkbox.
- title Human readable text that follows the checkbox
- checked Initial checkbox status. true/false.

Optional attributes:

- help (optional) If present, URL of the help HTML page.

Jelly code	Example

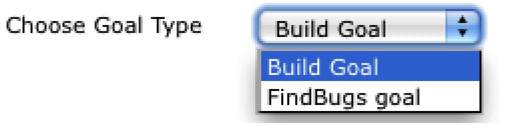
`<f:block>`
`  <table>`
`    <f:optionalBlock name="dynamic" title="Use existing dynamic view">`
`      <f:entry title="View drive">`
`        <f:textbox name="drive" value="${it.drive}"/>`
`      </f:entry>`
`    </f:optionalBlock>`
`  </table>`
`</f:block>`
	Unchecked the text box will not be displayed:
	
	Checked the text box is displayed
	

Select (drop-down menu)

Use an <f:entry> tag to enclose the normal select tag.

Jelly code	Example

```
<f:entry name="goalType" title="Choose Goal Type" field="goalType">
    <select name="goalType">
        <option value="buildGoal">Build Goal</option>
        <option value="findBugsGoal">FindBugs goal</option>
    </select>
</f:entry>
```



Select (drop-down menu) with model filled values

Basically the same as above. You need to define the following method in the descriptor of your Describable instance:

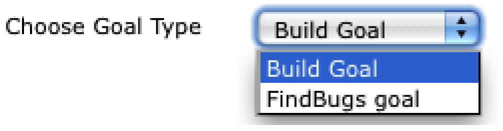
```
        public ListBoxModel doFillGoalTypeItems() {
            ListBoxModel items = new ListBoxModel();
            for (BuildGoal goal : getBuildGoals()) {
                items.add(goal.getDisplayName(), goal.getId());
            }
            return items;
        }
```

Then, use an <f:entry> tag to enclose the normal select tag.

Jelly code

```
<f:entry field="goalType" title="Choose Goal Type">
   <f:select />
</f:entry>
```

Example


Choose Goal Type
Build Goal
Build Goal
FindBugs goal