

Js Baseclasses

These API's are found on components that inherits from the baseclass



SmartBinding API

- getTraverseNodes
- getFirstTraverseNode
- getLastTraverseNode
- isTraverseNode
- getNextTraverseSelected
- getNextTraverse
- getTraverseParent
- getActionTracker
- connect
- disconnect
- setBindClass
- removeBindClass
- getBindClass
- getSelectionBindClass
- setSelectionBindClass
- getModel
- setModel
- load
- reload

DragDrop API

- copy
- move

SelectBindings API

- change

Transaction

- commit
- rollback
- startTransaction

jpf.Model Properties

- data
- caching

Js Objects

jpf.Model Methods

- toString
- register
- unregister
- getXml
- parse
- loadComm
- loadFile
- clear
- load
- reloadJmlNode
- reset
- extend
- insertRPC
- insertComm
- insertFile
- insert

Javeline IDEF's

- __WITH_DATABINDING
- __WITH_MODEL
- __WITH_CACHE
- __WITH_DRAGDROP
- __WITH_MULTIBINDING
- __WITH_RSB
- __WITH_IMPLICIT_XSLT
- __WITH_EXPLICIT_XSLT
- __WITH_CSS_BINDS
- __WITH_SORTING

SmartBinding Syntax

```
<j:smartbinding id="" model="" bindings=""
actions="" dragdrop="" >
  <j:model />
  <j:bindings />
  <j:actions />
  <j:dragdrop>
    <j:allow-drag select="" [rule] />
    <j:allow-drop select="" [rule] />
  </j:dragdrop>
</j:smartbinding>
```

Model Syntax

```
<j:model get="proclnstr" />
<j:model><data /></j:Model>
```

Bindings Syntax

```
<j:bindings id="">
  <j:rule select="XPath" />
  <j:rule select="XPath" mask="true:false:xx" />
  <j:rule select="XPath" default="literal" />
  <j:rule select="XPath" method="name" />
  <j:rule select="XPath" eval="expression" />
  <j:rule select="XPath">
    <xsl:if test="XPath">
      <xsl:value-of select="XPath" />
    </xsl:if>
  </Rule>
  <j:rule select="XPath">
    <!-- put your XSLT stylesheet here -->
  </j:rule>
  <j:load select="XPath" [sources] />
  <j:insert select="XPath" [sources] />
  <j:traverse select="XPath" [sorting] />
</j:bindings>
```

Actions Syntax

```
<j:actions id="" [data-update="deferred" rpc-
mode="realtime" defer-default={"add" | "update"}]>
  <j:rule select="XPath" set="proclnstr" />
  <j:add select="XPath"><data /></j:add>
</j:actions>
```

DragDrop Syntax

```
<j:dragdrop id="">
  <j:allow-drag select="XPath" copy-condition="event.ctrlKey" />
  <j:allow-drop select="XPath" target="XPath" action="list-append" copy-condition="event.ctrlKey" />
  <j:allow-drop select="XPath" target="XPath" action="tree-append" copy-condition="event.ctrlKey" />
  <j:allow-drop select="XPath" target="XPath" action="insert-before" copy-condition="event.ctrlKey" />
</j:dragdrop>
```

Sorting Syntax

```
<j:traverse select="XPath" sort="XPath" data-type={"string" | "number" | "date"} date-format="DDMMYYYY"
sort-method="" order={"ascending" | "descending"} case-order={"upper-first" | "lower-first"} />
```

Remote SmartBindings Syntax (multiclient data synchronization)

```
<j:remotesmartbindings socket="id">
  <TagName select="XPath" unique="XPath" />
</j:remotesmartbindings>
```

Jml Component Syntax

```
<j:Component smartbinding="" bindings=""
actions="" model="" select-bind="" select-
model="" bind="" dragEnabled="" dropEnabled=""
dragMoveEnabled="" actiontracker="">
  <j:smartbinding />
  <j:bindings />
  <j:actions />
  <j:dragdrop />
  <j:bind select="" [modifiers] />
  <j:action select="" [sources] />
</j:Component>
```

Processing Instructions

Model Instruction

```
model="id"
model="id.xpath"
model="#component:select"
model="#component:select.xpath"
model="#component:choose"
model="#component:choose.xpath"
model="#component:xpath"
model="<source>"
```

Get instruction

```
get="<source>|xpath|clear"
Gets result from a source. Then selects a subset using xpath. Clear sets wether the contents of the insertion point is cleared before appending the data.
```

Set instruction

```
set="<source>"
```

<source> syntax

Arguments *also valid for normal model syntax*

```
xpath:      xpath:/path/to/node
eval:       eval:5+10
method:     call:mName(xpath)
rpc:        rpc.comm.blah(xpath, call)
url:        url:http://example.com?x=xpath
cookie:     cookie:name.sub(xpath)
array:      ()
literal:    interpreted as javascript
```

Example: <j:load get="rpc.comm.getUserInfo(xpath:@id, getCache(), 'extrainfo')[data/user|1]" />

Js Objects

jpf.ActionTracker

Properties
realtime
hasChanged

jpf.ActionTracker

Methods
define
execute
addActionGroup
purge
reset
undo
redo

jpf.SmartBinding

Methods
load
initialize
deinitialize
addBindRule
addBindings
addActionRule
addActions
addDropRule
addDragRule
addDragDrop

jpf.XMLDatabase API

Lookup methods
getElementByld
getNode
getNodeByld
getDocumentByld
getDocument
getChildNumber
isChildOf
findHTMLNode
findXMLNode
getElement
getld
getModel
setModel
findModel

Dom Alterations
setTextNode
setAttribute
replaceNode
addChildNode
appendChildNode
moveNode
removeNode
removeNodeList

Utility methods
clearConnections
serializeNode
integrate
setNodeValue
getNodeValue
getInheritedAttribute
selectNodes
selectSingleNode
serializeXmlToXPath
unSerializeXmlToXPath
createNodeFromXPath
getXmlDocId
getBindXmlNode
getTextNode
getAllNodesBefore
getAllNodesAfter
getArrayFromNodelist
getDatalIsland
getXml
getObject