



AGETOR[®]

JSP Tag Library
User Guide

Contents

Contents	2
Preface	3
AGETOR JSP Tag Library	4
The valueof tag	5
The if tag	6
The choose tag	7
The for tag	8
The foreach tag	9
The folder tag	11
The session tag	13
The parameter tag	14
The link tag	15
The component tag	16
The datacomponent tag	18
The login tag	19
The barcode tag (optional)	20
Where to find more information	21

Preface

The AGETOR JSP tag library enables JSP programmers to use common IML tags in their JSP files. It contains custom tags, that enable web developers to add more dynamic functionality to their pages without needing much Java Code knowledge. These tags look like HTML-tags and are very powerful, though requiring a minimum of implementation effort.

This document is meant to be a reference to web developers, who want to develop their JSP pages in the same easy way as when developing IML files. It explains how to use all the different tags in the AGETOR JSP tag library, and illustrates each of them with examples.

We assume that you are familiar with JSP, and probably already use it. Even though you don't need to know IML to use the AGETOR JSP tag library, it is a good idea to get familiar with it. You can read about IML in the AGETOR IML user guide. Some Extraware features, as datacomponents and components, are used in the examples. Don't get frustrated if you don't know extraware, as it is not necessary to understand the examples. If you want to know more about Extraware, we recommend you read the Extraware Developer's Guide (EDG) and the different Extraware products' User Guides.

Notice that the current implementation of the AGETOR JSP tag library is based on the JSP 1.1 specification.

AGETOR JSP Tag Library

In the following we will describe every tag in the AGETOR JSP tag library. Every description contains a table explaining the use of the tag's attributes (if any) and one or more examples. If a tag contain nested tags inside it, it also includes a description, an attribute table and examples for each of them.

The attribute table contains 4 columns:

Attribute: the name of the tag's attribute.

Requ: indicates if this tag is required (✓) or optional.

Java: indicates if this attribute's value may be a Java statements (✓), or if it must be a fixed string.

Description: explains what this attribute does.

The AGETOR JSP tag library contains the following tags:

- valueof
- if
- choose
- for
- foreach
- folder
- session
- parameter
- link
- component
- datacomponent
- login
- barcode

To use any of these tags in your JSP page, remember to include the taglib directive:

```
<%@ taglib uri='inside.tld' prefix='inside' %>
```

Even though this tag library is meant to enable JSP programmers to use common IML tags, there are some differences between the JSP tags in this library and IML tags. Some IML tags are not supported by the AGETOR JSP Tag Library:

- print
- println
- set
- variable
- flush
- space
- tabs

Some other tags have changed their names:

- for-each
- value-of

And some tags in the AGETOR JSP Tag Library have more functionality than their corresponding IML tags (see the different tag descriptions in this document).

The reason for those differences are the limitations in JSP and IML respectively. Nevertheless the AGETOR JSP Tag Library provides you everything you need to make JSP documents as flexibly as you are used to with IML.

The `valueof` tag

This tag prints the value of a variable in the HTML page. Normally you can simply use the JSP syntax to do this operation: `<%= variable %>`

The `valueof` tag extends this functionality by adding the formatting of decimal numbers and date values, and should only be used in such cases. The date format syntax is the same as the one used in `java.text.SimpleDateFormat` (see its API Specification for a detailed description).

This tag contains the following attributes:

Attribute:	Requ:	Java:	Description:
name	✓	✓	The name of the variable.
decimal		✓	Number of digits to show on the decimal part of a number. The number is rounded.
todateformat		✓	The format to use when printing the date.
fromdateformat		✓	The format the date variable has. The default is the <code>java.util.Date</code> format (EEE MMM dd HH:mm:ss zzz yyyy).
condition		✓	If this condition evaluates to false, then this tag is simply ignored.

Examples:

Assuming `var` is the number 4.256, the following tag will print it out as 4.26:

```
<inside:valueof name='<%= var %>' decimal='2' />
```

Assuming `dat1` is the date `new java.util.Date(101, 1, 26)`, the following tag will print it out as 26.02.2001:

```
<inside:valueof name='<%= dat1 %>' todateformat='dd.MM.yyyy' />
```

Assuming `dat2` is the String 270200, the following tag will print it out as 27.02.2000:

```
<inside:valueof name='<%= dat2 %>' todateformat='dd.MM.yyyy' fromdateformat='ddmmyy' />
```

The `if` tag

This tag prints its body only if its `test` attribute evaluates to true.

This tag contains the following attribute:

Attribute:	Requ:	Java:	Description:
test	✓	✓	A boolean expression.

Example:

Assume that `ran` is a random number between 0 and 1. The following tag will only print its body when the random number is less than 0.2:

```
<inside:if test='<%= ran < 0.2 %>'>
    The random number is less than 0.2 -> <%= ran %>
</inside:if>
```

The choose tag

This tag allows you to choose one of several bodies, providing a condition for each of them. The first condition that evaluates to `true` will have its body printed out, and all the others will be ignored. If none of the conditions evaluates to `true`, the default body (`else` tag) will be printed out.

The `when` tag contains the following attribute:

Attribute:	Requ:	Java:	Description:
test	✓	✓	A boolean expression.

Example:

Assume that `ran` is a random number between 0 and 1. The following tag will print `too little` when the random number is less than 0.2, otherwise it will print the random number itself.

```
<inside:choose>
  <inside:when test='<%= ran < 0.2 %>'>
    too little
  </inside:when>
  <inside:else>
    <%= ran %>
  </inside:else>
</inside:choose>
```

The `for` tag

This tag allows iteration of its body. It starts at the `start` value and stops just before the `stop` value. The index used to count the iteration can also be used in the tag's body (`indexname`).

If `start < stop`, the `for` body is performed increasing from `start` to `stop`.
If `start > stop`, the `for` body is performed decreasing from `start` to `stop`.

When `start` equals `stop`, the body is NOT performed.

This tag contains the following attributes:

Attribute:	Requ:	Java:	Description:
<code>start</code>		✓	The first value of the iteration. If it is omitted, the default value 0 is used.
<code>stop</code>	✓	✓	The value where the iteration stops. This value will not be included in the iteration.
<code>indexname</code>			String variable containing the string representation of the iteration value. If this value is to be used as a number, it must be parsed first.

Example:

The following tag will print its body 3 times after each other (result: 1, 2, 3):

```
<inside:for start='1' stop='4' indexname='y'>  
    <%= y %>,  
</inside:for>
```

The foreach tag

This tag allows iteration of sequences. Supported types of sequence are `java.util.Vector`, `java.util.Hashtable`, `java.util.Iterator`, `com.sun.java.util.collection.Iterator` and `java.util.Enumeration`. It runs through the sequence of elements, performing its body one time for each element in the sequence. All elements in the sequence must be of the same type.

The sequence's iteration can be sorted by one or more attributes of the sequence's elements. For example, if the sequence has elements of the following type:

```
public class Element {
    private String name, secret;
    public double amount;

    public String getName() {
        return name;
    }
}
```

then the iteration can be sorted by name or amount, but not by secret, as it is a `private` variable and it does not have a `get` method.

This tag contains the following attributes:

Attribute:	Requ:	Java:	Description:
name	✓		The name of the variable holding an element from the sequence at each iteration.
type	✓		The full class name of the sequence's elements.
context		✓	If the elements of the sequence are entities or other objects that takes a <code>Context</code> (package <code>dk.bording.inside.container</code>) parameter, it can be supplied with this attribute.
sequence	✓	✓	The name of the variable containing the sequence.
indexname			String variable containing the string representation of the iteration value. If this value is to be used as a number, it must be parsed first.
sort		✓	The names of the attributes to sort the iteration by. The attributes must be separated by comma. Default sorting is ascending. To sort in descending order, add <code>desc</code> after the attribute's name. The given attributes must be <code>public</code> in their class or have a <code>get</code> method.
start			If you do not want to go through the whole sequence, you can set this attribute to start at a specific index.
stop			If you do not want to go through the whole sequence, you can set this attribute to stop before the specific index. This index will not be included in the iteration.

Example:

Assuming that the `Vector v` contains the following elements:

```
Vector v = new Vecor();
v.add(new Element("a", 1.0, "s1"));
v.add(new Element("c", 6.0, "s2"));
v.add(new Element("a", 7.0, "s3"));
v.add(new Element("c", 4.0, "s2"));
```

the following tag will print the their names in descending order, then ordered by their amounts:

```
<inside:foreach name='elem' type='Element' sequence='<%= v %>'
    indexname='i' sort='name desc, amount'>
```

```
<%= elem.name %>, <%= elem.amount %><br/>  
</inside:foreach>
```

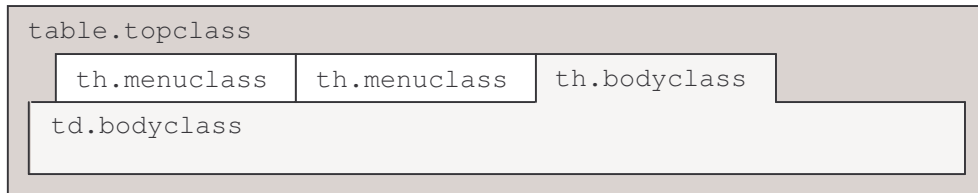
and the result will be:

```
c, 4.0  
c, 6.0  
a, 1.0  
a, 7.0
```

The folder tag

This tag creates a HTML folder, allowing your web page to have layers. Every layer is called a tab.

The appearance of the folder is determined by the `topclass`, `menuclass` and `bodyclass` values. These values correspond to the classes within the JSP page's style (whether it is in the page itself or in a stylesheet). The folder tag generates a HTML table, so these classes should format the table, and its cells, as shown in the following picture:



The folder must be inside a HTML form to work. Whenever you select a tab in the folder, the input `<folder name>_folder_selection` will be submitted, with the selected tab's name as value. The page itself is then requested again, and the selected tag's content is shown.

This tag contains the following attributes:

Attribute:	Requ:	Java:	Description:
name		✓	The folder's name.
topclass		✓	The style class of the folder's <code><table></code> tag.
menuclass		✓	The style class of the folder's non-selected <code><th></code> tags.
bodyclass		✓	The style class of the folder's selected <code><th></code> tag and all its <code><td></code> tags.

The tab tag contains the following attributes:

Attribute:	Requ:	Java:	Description:
name	✓	✓	The tab's name. When the tab is clicked, this name is the value of the form input <code><folder_name>_folder_selection</code> .
title		✓	The string that is shown on the tab (do not use this with <code>img</code>).
img		✓	The image that is shown on the tab (do not use this with <code>title</code>).
selectedimg		✓	The image that is shown on the selected tab (use this with <code>img</code>).

Example:

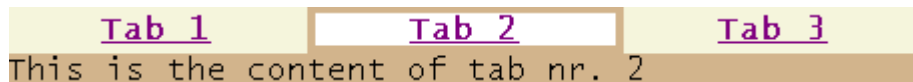
If we assume that the JSP page has the following style:

```
<style>
    table.topclass {
        font-family: "lucida console";
    }
    th.menuclass {
        background-color: beige;
        width: 150px;
    }
    th.bodyclass {
        border: solid tan;
        width: 150px;
    }
    td.bodyclass {
        background-color: tan;
    }
</style>
```

The following tag will produce a folder with 3 tabs. As you can see, the `tab` tag can be iterated using a `for` tag or a `foreach` tag:

```
<form name='form1'>
  <inside:folder name='MyFolder' topclass='topclass' menuclass='menuclass'
    bodyclass='bodyclass' formid='form1'>
    <inside:for start='1' stop='4' indexname='i'>
      <inside:tab name='<%= "tab"+i %>' title='<%= "Tab "+i %>'>
        This is the content of tab nr. <%= i %>
      </inside:tab>
    </inside:for>
  </inside:folder>
</form>
```

The result will be:

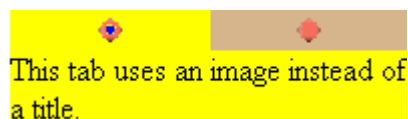


The next tag will produce a folder with 2 tabs that use images instead of titles. First we show the style:

```
<style>
  table.topclass2 { width: 200px; }
  th.menuclass2 { width: 100px;
    background-color: tan; }
  th.bodyclass2 { width: 100px;
    background-color: yellow; }
  td.bodyclass2 { background-color: yellow; }
</style>
```

```
<form name='form1'>
  <inside:folder name='folder2' topclass='topclass2' menuclass='menuclass2'
    bodyclass='bodyclass2'>
    <inside:tab name='deftab' img='/ifw/images/nom/prik.gif'
      selectedimg='/ifw/images/selected/prik.gif'>
      This tab uses an image instead of a title.
    </inside:tab>
    <inside:tab name='othertab' img='/ifw/images/nom/prik.gif'
      selectedimg='/ifw/images/selected/prik.gif'>
      So does this one.
    </inside:tab>
  </inside:folder>
</form>
```

The result will be:



The session tag

This tag allows JSP pages and servlets (e.g. IML pages) to share a session, and through this session's data they can communicate with each other.

Many of the tags in the AGETOR JSP tag library need this tag to work, otherwise a `JSPException` is thrown. Therefore we recommend you to add this tag to all your JSP pages that use the AGETOR JSP tag library.

The session tag should always be placed at the top of the JSP page, just after the taglib directive.

Example:

```
<%@ taglib uri='inside.tld' prefix='inside' %>
<inside:session />
```

The parameter tag

This tag defines a variable to a servlet or extracts the variable's value from the URL parameters of the form `http://mypage.jsp?<parameter_name>=<parameter_value>`. If the URL does not contain the parameter, a default value can be used.

This tag contains the following attributes:

Attribute:	Requ:	Java:	Description:
name	✓		The variable's name. This name must match the name of the URL parameter it gets its value from.
type	(✓)		The variable's type (primitive types are not supported).
default		✓	Determines a default String value for the parameter and can therefore only be provided for Strings. If you want an empty string as default, do not use <code>default=''</code> , but use <code>default='EMPTY'</code> uppercase only instead, otherwise it will not work.
object		✓	Determines a default object value for the parameter. Unlike default, the parameter's variable must have been instantiated in advance.
isnew			Indicates whether to declare the variable for you or use an existing one. If you already have a variable of the same name as this parameter, you should set this attribute to <code>false</code> . The default value is <code>true</code> . The <code>object</code> attribute implicitly sets this attribute to <code>false</code> .

Example:

The following tag will extract the string `your_name` from the URL. If it's not found, the parameter value will be an empty string:

```
<inside:parameter name='your_name' type='String' default='EMPTY' />
```

Another way of having an empty string as default, is to declare the variable yourself, and use its value (the empty string) as default:

```
<% String x = ""; %>
<inside:parameter name='x' type='String' isnew='false' default='<%= x %>' />
```

The following tag will extract the integer number `number` from the URL, and save it in the `number` variable. If the parameter is not found in the URL, the variable's value (36) will be used instead:

```
<% Integer number = new Integer(36); %>
<inside:parameter name='number' type='Integer' object='<%= number %>' />
```

If you omit the `object` attribute, and use `isnew='false'` instead, the default value will be 0, and not 36:

```
<% Integer number = new Integer(36); %>
<inside:parameter name='number' type='Integer' isnew='false' />
```

The `link` tag

This tag corresponds to the HTML `<a>` tag, except that it correctly URL-encodes the link and its parameters for you. Observe that the `parameter` tag inside a `link` tag differs from the previously mentioned `parameter` tag, as it *provides a value* for a URL parameter, instead of *extracting its value* from the URL.

This tag needs JavaScript 1.1. or newer to work properly.

This tag contains the following attributes:

Attribute:	Requ:	Java:	Description:
name	✓	✓	The link's URL.
target		✓	The target frame where to display the link. If none is given, the frame of the JSP page itself will be used.
type		✓	Determines if it is an Inside link (<code>'inside'</code> , meaning that the actual servlet to link to is defined in <code>servlets.cfg</code>) or any other kind of link (<code>'other'</code>). The default is <code>'inside'</code> .
autoload		✓	If this parameter is set to true, this link will load itself without the user clicking on it. The default is false.
time		✓	The amount of time (in seconds) to wait before autoloading the link. If autoload is set to false, this parameter is meaningless. The default is 0 (zero).

The `parameter` tag contains the following attributes:

Attribute:	Requ:	Java:	Description:
name	✓		The parameter's name.
value		✓	A string containing the parameter's value.

Example:

The following tag will print a link to Bording's homepage with the `user_age` parameter containing the value 25:

```
<% Integer age = new Integer(25); %>
<inside:link name='http://www.bording.dk' target='_blank'>
  Bording's homepage.
  <inside:parameter name='user_age' value='<%= age.toString() %>' />
</inside:link>
```

The component tag

This tag allows you to use AGETOR components directly in your JSP page. Any components that are available for IML pages, can also be used in JSP.

The `property` tag allows you to set values for the component's properties, and the `section` tag allows you to overwrite the component's sections.

The `component` tag also has an `invoke` tag, that *can only be called from inside a section tag*. The `invoke` tag, as its name says, invokes the given section. We do not encourage you to use this tag, though, as it can cause serious problems. If you are going to use it, remember not to call it from the called section itself, as it will cause an infinite loop.

This tag contains the following attribute:

Attribute:	Requ:	Java:	Description:
name	✓		The component's logical name. This name can be found in the <code>components.cfg</code> file.

The `property` tag contains the following attributes:

Attribute:	Requ:	Java:	Description:
name	✓		The name of the property.
value	✓	✓	The property's value (must be a String, that will then be parsed to its actual type).

The `section` tag contains the following attribute:

Attribute:	Requ:	Java:	Description:
name	✓		The name of the section to be overwritten.

The `invoke` tag contains the following attribute:

Attribute:	Requ:	Java:	Description:
name	✓		The name of the section to be invoked.

Example:

The following tag will print the Extraware toolbar component:

```
<inside:component name='extraware.toolbar'>
</inside:component>
```

The result will be:



The following tag will print the same toolbar component, but with its `undo` property set to `true` (it is `false` by default), and its `space` section replaced by a section containing only an empty table cell:

```
<inside:component name='extraware.toolbar'>
  <inside:property name='undo' value='true' />
  <inside:section name='space'>
    <td width="17" height="29">
      </td>
  </inside:section>
</inside:component>
```

The result will be:



The datacomponent tag

This tag allows the JSP page to access AGETOR datacomponents, in the same way as in IML files.

This tag contains the following attribute:

Attribute:	Requ:	Java:	Description:
name	✓		The datacomponent's logical name. This name can be found in the <code>datacomponents.cfg</code> file.

Example:

The following tag will make the Extraware `menu_global` datacomponent available from the JSP page:

```
<inside:datacomponent name='menu_global' />
```

The datacomponent can then be used as in this example:

```
<% Vector menus = menu_global.listMenuGroups(""); %>
```

The login tag

As you know, security control can be made through the AGETOR Broker. This tag allows you to check user login through the broker.

If the login is successful, the body of the `loginok` tag will be evaluated, otherwise the body of the `loginerror` tag will be evaluated instead.

This tag contains the following attributes:

Attribute:	Requ:	Java:	Description:
user		✓	The user name. If this attribute is not given, its value will be extracted from the session data.
password		✓	The user's password. If this attribute is not given, its value will be extracted from the session data.
env		✓	Determines the environment of the broker. This has normally been set up in the <code>http</code> setting. If this attribute is not given, its value will be extracted from the session data.

Example:

The following tag will print "Login Ok!" if the login is successful, otherwise it will print "Login failed!":

```
<inside:login user='<%= user %>' password='<%= pswd %>' env='<%= extraware %>' >
  <inside:loginok>
    Login Ok!
  </inside:loginok>
  <inside:loginerror>
    Login failed!
  </inside:loginerror>
</inside:login>
```

The barcode tag (optional)

The barcode tag produces printable barcodes on your web page. It supports the following types of barcodes:

- Ean8
- Ean13
- Ean128
- Code39

There are two implementations you can use in the library. Dynamically generated barcode images, or image puzzles consisting of individual images that will make up a barcode. See the Barcode documentation for instructions on creating images dynamically.

This tag contains the following attributes:

Attribute:	Requ:	Java:	Description:
code	✓	✓	The code for the image
type	✓	✓	The barcode type
height		✓	The height of the barcode in pixels (dynamically generated images only)

Example:

```
<% String partNo1 = "4711"; %>
<% int height = 30; %>
<inside:barcode code='<%= partNo1 %>' type='code39' height='<%= height %>' />
```

will result in this barcode image (using dynamic image generation):



Where to find more information

If you have questions about the AGETOR JSP tag library, or other AGETOR products, you can look for answers in the AGETOR Downloadcenter (<http://www.agetor.com>), or write to AGETORsupport@bording.dk.