ITC JSP Taglib Technical Reference
Ver. 2.3

TABLE OF CONTENTS

Setting up the taglib environment

The taglib definition file (itctaglib.tld) is in the META-INF directory in the itcworks.jar file. It is not necessary to put the definition for these tags in the web.xml file (as in earlier Java Server Pages (JSP) versions). The JSP URI should be defined in your page as follows:
<%@ taglib uri="http://www.i-techcorp.com/jsp/ext" prefix="itc" %>

Control Naming Conventions

Using the ITC tags eliminates the need of having to inherit your Java form objects from a specified base class in order to have the object properties loaded from a form submission. The ItcServletRequestToBean class moves submitted form data to any Java object or object graph that has matching properties. The required naming conventions are as follows:

- You must use the **<itc:input>**, **<itc:select>** (for list/combo boxes) and **<itc:iterator>** (for collections) in conjunction with the **<itc:form>** tag.
- The name attribute on the control (except controls inside the **<itc:iterator>** tag) have the form: **ClassName.propertyName** where **ClassName** is the name of the Java class that will receive the submitted form values and property name is the property on that class to receive the value. For example: If you had an Address object on an input form, your control name might look something like: <itc:input name="Address.city" type="text" etc…./>.
- If you are inside an **<itc:iterator>** tag, then the control name follows this format: **ClassName$<iteractionName>.propertyName** where **iterationName** is the same as the "name" attribute on the **<itc:iterator>** tag. The following example demonstrates displaying a list of Person objects on a form.
  <itc:iterator name="*person*" collection="people">
    <itc:input type="text" name="Person$*person*.firstName"/>
    <itc:input type="text" name="Person$*person*.lastName"/>
      etc…
   </itc:iterator>

Using this naming convention with the ItcSevletRequestToBean allows any number of different object types to be used on an input form. Inside your servlet, struts action (or whatever Java mechanism you are using), you just invoke the static method toBean as follows:
ItcServletRequestToBean.toBean(request, objInstance); where **request** is an instance of the ServletRequest and **objInstance** is an instance of the bean or object graph you're loading.

The ${macroName} substitution

The ***${macroName}*** is a server side runtime evaluation that is first performed before the attribute is used in a tag lib call. This provides tag lib attribute values to by dynamically resolved. Take for example an <itc:if> tag that compares a value from a bean to some value provided by the value attribiute:
<itc:if name="checkingAccount" property="balance" nop=">" value="100.00">
  <strong>Account gets interest</strong>
</itc:if>

The value attribute can take a string literal but you might want the value to be retrieved from a resource bundle or some other bean object. The value attribute could be defined as follows: value="***${bank.interestThreshold}***" where ***bank.interestThreshold*** is a resource bundle key or value="***${bankBean.minInterestBal}***" where ***bankBean*** is a Java bean in the pageContext, request or session object and ***minInterestBal*** is a property holding the minimum balance required for an interest checking account.

The macro name can represent one of the following values and is shown in the order of evaluation precedence:

- Any property defined in the Java System properties or a loaded resource bundle. ex. ${user.home}, ${user.name}, ${my.bundle.key} etc…

- Any bean property defined in the pageContext, request or session context or any map based object with the form mapName.keyName where mapName is a map based object and keyName is the map key.
  Examples: ${account.balance} where balance is a getter method on an object referenced by account.
  ${request.name} where request is the HttpServletRequest object and name is either a parameter name or an attribute value.
  ${session.attrKey} where session is the HttpSession object and attrKey is an attribute key of the session

  ${pageContext.attrKey} where pageContext is the JSP pageContext and attrKey is the pageContext attribute key

Macro names can be concatenated to string literals or other macro names to resolve more complex situations. The following example shows a Map object called categories that has value names of other map objects for each user interest like sports, travel, cooking etc… The second macro refers to a bean named user which has the property getInterestName(); Let's say that the interests key returns the value of sports(a map of sports names) and interestName returns a value of baseball. The following expression: ***${categories.interests}.${user.interestName}*** would resolve to "sports.baseball" where sports is a map object and baseball is the map key containing some boolean value. The ***exists*** attribute then uses the value "sports.baseball" as it's test.

```
<itc:if  exists="${categories.interests}.${user.interestName}">
  <itc:write name="user" property="name"/> Likes <itc:write name="user" property="interestName"/>
</itc:if>
```

Please refer to the junitDemo.jsp file for more examples.

Example code

Please refer to the junitDemo.jsp and test.itc.jsp.actions.JUnitDemoAction class in the JUnitForItc source zip for the taglibs used in context. Each tag is thoroughly demonstrated in the junitDemo.jsp file.

Technical Reference

Each tag defined has the following format: The **Tag:** (which is the tag name), the **Description** and the **Attributes** list. Each attribute is listed with the required/optional requirement and the description of what it does.

Tag: <itc:if>:

Description: The ***<itc:if>*** tag provides conditional processing. It is a body tag and if the test condition is true, the body content of the tag will be submitted to the browser. The ***<itc:if>*** tag can also be used in conjunction with the ***<itc:elseIf>*** and ***<itc:else>*** tags providing the JSP developer standard if..else if… else natural logic. These tags can be nested inside each other as well. The three aforementioned tags are cooperating tags. Just like a standard programming language, the ***<itc:elseIf>*** and ***<itc:else>*** tags have to have a preceding ***<itc:if>*** tag.

Attributes:

name – **required**: The name attribute defines the name of the object that has been put in the pageContext, session or request object that will supply the value to test.

propert*y* ***– required: (if method attribute not specified)*** The property attribute names the bean property or a map key depending on the object referenced by the ***name*** attribute. The property name is expected to be a standard getXxx() method with no parameters on a bean.

method ***– required if property attribute not specified:*** The method attribute allows bean methods that do not start with property standard "get" to be executed. The return value is assumed to be a String or an Object that can be rendered as a string. The object's toString() method will be invoked by the tag. The method may also take parameters which are defined by the ***parameters*** attribute.

parameters - *optional*: The parameters attribute takes a comma separated list of parameter value(s) that correspond to the method signature of the method you are executing.

exists – *optional:* This performs a simple test for the existence of an object. This attribute names a bean property or one of: pageContext, request or session attributes that return an Object of some type. If the object exists, the body of the tag is sent to the browser.

notExists – *optional* This is converse of exists.

op – *optional* : The ***op*** attribute defines the logical operator to use for string tests. The allowed values are: = (equal),!= (not equal),< (less than),> (greater than),<= (less than equal),>= (greater than equal). If the op attribute is omitted, the = operator is assumed.

nop – *optional:*	The ***nop*** attribute (numeric operation) defines the logical operator to use for numeric tests. Unlike the ***op*** attribute which can be omitted for equal tests, the ***nop*** attribute must be specified for all numeric tests. The values are the same as defined for the ***op*** attribute.
value **– *required:***	The value attribute is the value you are testing for. It can be a literal value or ***${macroName}*** as described above. The value "null" may also be specified for null tests.
test **–** *optional:*	The test attribute defines a java boolean expression that is used in place of the name, property or method attributes.
scope **-** *optional:*	The scope attribute specifies the location of the bean or map object. Must be one of: *page*, *session* or *request*

Tag: <itc:elseIf>

This tag works in conjunction with the ***<itc:if>*** to provide if .. else if logic processing.

Attributes:
The attributes for this tag are the same as the ***<itc:if>*** tag

Tag: <itc:else>

Description: This tag works in conjunction with the ***<itc:if>, <itc:elseIf> tags*** to provide if .. else if .. else logic processing. It is invoked when no prior ***<itc:if>*** or ***<itc:elsIf>*** was met.

Attributes:
This tag has no attributes.

Tag: <itc:excludeIf>

Description: This tag is a convenient standalone NOT **if** condition. If the test condition is true, the body of the tag is skipped (not sent to the browser).

Attributes:

The attributes for this tag are the same as the ***<itc:if>*** tag

Tag: <itc:simpleIf>

Description: The ***simpleIf*** tag outputs the value of an expression or a string if the condition is true. This is handy when you want to assign a value like a CSS class type based on a condition.

Attributes:

All of the attributes defined in the **<itc:if>** tag are supported as well as the following additions:

result **– required:** The result attribute can be a string literal, a ***${macroName}*** (described above), or a <%=expression %>. If the test is true the value for this attribute is output.

Tag: <itc:simpleIfElse>

Description: The ***simpleIfElse*** tag works just like the ***simpleIf*** tag but adds a false result output string if the condition is false.

Attributes:

All of the attributes defined in the **<itc:if>** tag are supported as well as the following additions:

trueResult **– required:** The value to output if the test is true. See the attribute description for the *result* attribute described above in the ***simpleIf*** tag for its allowed values.

falseResult **– required:** The value to output if the test is false. See the attribute description for the result attribute above in the ***simpleIf*** tag for its allowed values.

Tag: <itc:write>:

Description: The write tag is used to display the contents of a Map entry, ItcDataObject entry, a bean property or a bean method. This differs from the struts **<bean:write>** tag in that you can display map entries for the specified key or invoke methods on any object. These methods don't have to start with "get" and may also take parameters.

Attributes:

name – **required:** The name attribute defines the name of the object that has been put in the page, session or request object.

p*roperty* **– *required: (if method attribute not specified)*** The property attribute names the bean property or a Map key depending on the object referenced by the ***name*** attribute. The property name is expected to be a standard getXxx() method with no parameters.

method **– *required: (if property attribute not specified)*** The method attribute allows bean methods that do not start with “get” to be executed. The return value is assumed to be a String or an Object that can be rendered as a string. The toString() method will be invoked by the tag. The method may also take parameters which are defined by the ***parameters*** attribute.

parameters **-** *optional*: The parameters attribute takes a comma separated list of parameter value(s) that correspond to the method signature of the method you are executing.

format – *optional:* This attribute specifies a data format string for formatting raw data values. These format masks have preset constants for numbers with decimals and commas, money, zip codes ssn numbers etc…The format values are defined in the javadoc for the class com.itc.utl.ItcEdit. To use one of the pre-defined masks defined as static constants in the ItcEdit class enclose the constant name inside the ${}. Format=”${MONEY}” for example would use the predefined MONEY mask as specified in ItcEdit class. Please refer to the javadoc for the format mask strings that are available.

*scope**-** optional**:*** The scope attribute specifies the location of the bean or map object. Must be one of: *page*, *session* or *request*

Tag <itcForm>

Description*:* The ***<itcForm>*** tag is used in conjunction with the **<itc:select>** and **<itc:input>** tags to automatically emit javaScript for form validation. This tag insures that the form will only be submitted when any validation on input control type numeric and decimal or a required select (for list and drop down boxes) have passed. . The current validations supported are **required** for **<itc:input>** and **<itc:select>** tags and numeric and decimal types for **<itc:input>** tags. Please refer to the descriptions in the **<itc:select>** and <**itc:input>** tags for the specific attribute values.

Form validation and submission

To invoke the javascript validation, you must execute one of the following javaScript functions that are created by this tag when any **<itc:input>** or **<itc:select>** tag is used with the above described attributes/types:

validate(formObject) : Use the validate function when the button is a submit button. A typical scenario uses the following form:
<itc:input” type=”submit” name=”save” value=”Save” onClick=”return validate(this.form)”/>

For input types that are of type button i.e. type=”button”, the following functions can be used:
submitForm(formObject, sUrl) where **formObject** is the form reference i.e. this.form and **sUrl** is a string parameter representing the URL the form submission uses.

<itc:input” type=”button” name=”save” value=”Save” onClick=”return submitForm(this.form, ‘/myAction.do’)”/>

Use this function when using html frames and a frame target is needed.
submitFormWithTarget(formObject, sUrl, sTarget) where **sTarget** is a string parameter naming the targeted frame.

Attributes:

The <itc:form> supports the same attributes as the html form element does. Please refer to the html form element documentation for their values.

Tag: <itc:select>

Description: The select tag defines a list or drop down list. It generates the html select and option tags based on values defined in a Map, List or an ItcDataObjList. When using a Map based object, the Map key represents the display value and the key value is sent on the form submission. When using a List collection, the toString() is invoked on each List element. The display and form submission values are the same. If an ItcDataObjList is used then each ItcDataObject will have a "DispValue" and a "Value" entry.

Attributes:

name **–** ***required:*** The **name** attribute names the list/drop down control on the form. It should be in the format as described in the control naming conventions above.

beanName **–** *optional:* (if the ***collection*** attribute is not defined). The **name** attribute names the list/drop down control on the form. It should be in the format as described in the control naming conventions above.

property **–** ***required:*** *(*if th*e* ***name*** is used) The ***property*** attribute names the property on the bean that returns a Map, List or an ItcDataObjList.

collection **–** *optional:* (if name and property omitted) The name of the Map or ItcDataObjList stored in the page,session or request object

noSelectItem – *optional:* The value of this attribute is guaranteed to be the first item in a drop down combo box. This value is not defined in the collection that loads the drop down options and is used to detect a non selection. The default value for this selection is an empty string but a different value may be specified using the noSelectItemValue attribute.

noSelectItemValue – optional: This attribute is only needed if the value associated with the noSelectItem needs to be different than the default empty string.

required - *optional:* The required attribute can have the value "true" or it can be a resource bundle key in the from **itcrb.yourKeyName** where **itcrb.** Is the recognized bundle prefix and **yourKeyName** is the resource bundle key. The resource bundle key method allows you to name the field which is incorporated in the alert box message when the field is blank. If the value "true" is used, then the generic message "Please make a Selection" is used. You may also override this message by putting your own message in a resource bundle with the bundle key set to itc.selectionRequired. You will need to use this approach if internationalization is required in your application. If you are using the field name or bundle key as the value for the required attribute, then use the %1 place holder in your string. The properties file entry would look something like itc.selectionRequired=The %1 field requires a selection.

selected – *optional* This value typically names a bean property or a Map key that defines the value that should be set for a list or combo box. If a bean property is needed then the attribute should use the form beanName.propertyName where beanName is defined in the page, request or session objects.

selectedValues **–** *optional:* The selectedValues attribute is the name of a Map object stored in the page,session or request object that has an key entry for each value selected in a list or a drop down control.

size **–** *optional:* Same as the html size attribute for the select tag

multiple **–** *optional:* If the multiple is set to true, then the List control allows multiple selections

styleClass **–** *optional:* This attribute sets the CSS class. Because, class is already defined in Java, we have to use styleClass.

The following attributes are also supported as defined in the html select control: accesskey, align, disabled,onFocus,onBlur,onChange,onClick,style,tabIndex and title. Please refer to the html documentation for these values.

Tag: <itc:input>

Description: The **<itc:input>** tag can be used to define html data entry controls like text, textarea, buttons, etc … This tag will emit java script when the required, numeric and decimal types are defined for form validation. NOTE! This tag MUST be used in conjunction with the ***<itc:form>*** tag.

Attributes:

name – ***required***:. This names the input control. It should be in the format as described in the control naming conventions above.

required - *optional:* The required attribute can have the value “true” or it can be a resource bundle key in the from **${propertyName}** where propertyName is the resource bundle key. The resource bundle key method allows you to name the field which is incorporated in the alert box message when the field is blank. If the value “true” is used, then the generic message “This field cannot be blank” is used. You may also override this message by putting your own message in a resource bundle with the bundle key set to itc.entryRequired. You will need to use this approach if internationalization is required in your application. If you are using the field name or bundle key as the value for the required attribute, then use the %1 place holder in your string. The properties file entry would look something like itc.entryRequired=The %1 field cannot be blank.

type – ***required:*** This attribute defines the input control type and takes the same values as the html type attribute but also adds the following new types:

numeric (only allows digits 0 – 9)

decimal (only allows digits 0 – 9 and one decimal point)

NOTE! You may override the alert message in the same manner described above for the required attribute by placing the following bundle keys in a properties file:

numeric override is itc.notNumeric
decimal   override is itc.notDecimal.

format – *optional:*	Please refer to the description defined in the **<itc:write>** tag above.

The list of attributes as defined by the html input is supported here as well. Please refer to that documentation for their values.

Tag: <itc:iterator>

Description: The ***<itc:iterator>*** supports object iteration over the following object types:

- Any Collection based object
- Any Map based object
- A HashTable
- Any Array
- An Enumeration Object
- An Iterator object

You can retrieve the iterator index from the pageContext object using the attribute key "iter" + the iterator name. (see name attribute below). Ex. If I have an iterator named people then pageContext.getAttribute("iterpeople") would retrieve the current iteration index.

Attributes:

name **– *required:***	This names the object returned by each collection iteration. This name can then be referenced by other tags (such as ***<itc:write>*** etc …) that need access to the object.
collection **–** *optional*	(if bean attribute omitted) This names one of the supported collection objects (see above). This object must be placed in the page, session or request object.
bean **–** *optional*	(if collection omitted) This attribute names a bean that has a property that will return one of the supported collection objects (see above). This object must be placed in the page, session or request object.
property **–** *optional*	(if collection is specified) This names the property on the bean that returns a collection object.

offset – optional: The starting offset (zero based) from the start of the collection to start the for iteration. The default if omitted is zero

count – *optional:* The number of iterations to perform. The default is all iterations.

break – optional If this attribute is specified then the iterator will terminate after the current iterator body completes. The value specified should be "true".

Tag:<itc:debug>

Description: This tag dumps the contents of the request headers, request parameters, request attributes, session attributes, applications attributes and page context attributes. If the debug tag is placed in a JSP page with the ***show*** attribute omitted then nothing is output on the page until a request parameter contains the string itcdebug=[all | std] , headers, params, session, page]. See attributes below for a description on the options. If the ***show*** attribute is specified, then the specified objects are always displayed.

Attributes:

show **–** *optional:* if specified the debug tag will always dump the requested values at the location this tag is placed on the page. The allowed values are as follows and can be mixed in a comma delimited manner:

all - display all servlet and JSP object values which includes object types listed in the description listed above.

std - display only the http servlet request headers and request parameters

headers - display only the http servlet request headers

params - display only the http servlet request parameters

req - display only the request attribute names and object values

session - display only http session attribute names and object values

page - display only the page context and application attribute names and object values

Tag: <itc:content>

Description: This tag retrieves content from outside the JSP. This tag is useful when the JSP page must include html content created by other external processes (like content managers). The content tag can grab files on the local file system or it can be an HTTP URL from a website. The referenced content is then included in the output stream at the location of the content tag. Please see the junitDemo.jsp for examples of this tag.

Attributes:

contentLoc – ***required***: This attribute names the location on a file system or an HTTP URL. If the value for this attribute needs to come from a resource bundle use the for itcrb.your_bundle_key where your_bundle_key is the name of the property.

contentName ***– optiona*l** if the ***contentLoc*** attributute is an HTTP URL that includes the name of the resource to be retrieved otherwise it is the name of the resource to be retrieved.

Tag: <itc:counter>

Description: This tag names a counter variable that can be incremented or decremented. The variable can be used as an index to an array or for fixed iteration counts.

Attributes:

name – **required**	This names the counter variable
initial – optional	This gives the counter an initial starting value. If omitted, zero is the default value.
max – ***required***	This sets the upper or lower boundaries that terminates the count
step – optional	The value that each count is incremented or decremented by. The default is 1.
dec – optional	If true the counter is decremented else it is incremented

Tag: <itc:messages>

Description: This tag displays any messages placed in either the pageContext, request or session object by a java controller process. The Java controller creates an ItcMessageList object and adds ItcMessage objects to it. When all messages have been added, the save method on the ItcMessageList object is invoked. ItcMessage objects are grouped in three categories: ***info***, ***warning*** and ***error***. These message types can be associated with corresponding cascading style sheet classes using the attributes described below. The code snippet below shows the java action process

```
ItcMessageList msgList = new ItcMessageList();
msgList.add( new ItcMessage( ItcMessage.ERROR, "my error message" ));
msgList.add( new ItcMessage( ItcMessage.WARNING, "my warning message" ));
msgList.add( new ItcMessage( ItcMessage.INFO, "my info message" ) );
list.save( request );
```

Attributes:

infoClass – *optional:* Associates an ItcMessage.INFO type with the css class name specified

warningClass – *optional*	Associates ItcMessage.WARNING type with the css class name specified
errorClass – *optional*	Associates ItcMessage.ERROR type with the css class name specified
showOnly – *optional*	This attribute acts as a filter and is a comma separated list of category types to display. It can be a combination of info,warning and error. This allows the tag to be placed in multiple locations in the JSP page and show only the message types specified. If this attribute is omitted, then all messages in the ItcMessageList are displayed.

Tag:<itc:rb>

Description: This tag gives the JSP page Java resource bundle support.

Attributes:

load – optional:	The name of the resource bundle to load. This is based on the Locale associated with the JVM or HttpSession
merge – optional:	merges all keys in the loaded bundle into one properties file (the default)
key – optional:	only when load attribute is used) retrieves the value of the resource bundle key and writes it to the output stream
name – *optional:*	when merge is not on. Names the bundle to use for the ***key*** attribute specified