Investment Intelligence Systems Corporation

JSheet Java Server Pages Custom Tags

Version 1
November 2002

Copyright Information

Disclaimer of Warranties

The software and users manuals are provided "as is" and without express or limited warranty of any kind by either IISC or anyone who has been involved in the creation, production, or distribution of the software, including, but not limited to the implied warranties of the merchantability and fitness for a particular purpose. The entire risk as to quality and performance of the software and users manuals is with you. Should the software and users manuals prove defective, you (and IISC or anyone else who has been involved in the creation, production, or distribution of the software) assume the entire cost of all necessary servicing, repairs, or correction.

Some states do not allow the exclusion of implied warranties, so the above exclusion may not apply to you.

Limitation of Liability

In no event will IISC or any other person involved in the creation, production, or distribution of the software be liable to you on account of any claim for any damages, including any lost profits, lost savings, or other special, incidental, consequential, or exemplary damages, including but not limited to any damages assessed against or paid by you to any third party, arising out of the use, inability to use, quality, or performance of such software and users manuals, even if IISC or any other such person or entity has been advised of the possibility for such damages, or for any claim by any party.

In addition, IISC or any other person involved in the creation, production, or distribution of the software shall not be libel for any claim by you or any other party for damages arising out of the use, inability to use, quality, or performance of such software and users manuals, based upon principals of contract warranty, negligence, strict liability for the negligence of IISC or other tort, breach of any statutory duty, principals of indemnity or contribution, the failure of any remedy to achieve its essentials purpose, or otherwise. Some states do not allow the limitation or exclusion of liability for incidental or consequential damages, so the above limitation may not apply to you.

Introduction

About This Manual

This reference manual provides information about including JSheet custom tags in your JSP (Java Server Pages) files. When you include these custom tags, dynamic HTML is generated in your application programmatically via Java.

This Manual contains this Introduction and three chapters.

Chapter 1: Requirements & Component Overview; lists the system requirements, gives an overview of the components and covers deploying the JSheet JSP Custom Tags.
Chapter 2: Getting Started; describes the environment in which you include .jsp files
Chapter 3: JSheet JSP Custom Tags, an alphabetical reference to each custom tag.

Before You Begin

This manual assumes that you have a basic understanding of HTML tags and attributes and know how to build a web page. Before you can implement JSP pages in your web application, you must have installed:

- JSheet Server.
- A web server/JSP/Servlet engine suite that supports JSP1.1 and servlets 2.2.

Some web servers have JSP support built-in. Other web servers must add JSP support. Some JSP engines include a web server. Most web servers that understand JSP look for a specific filename extension. Typically, any filename that ends in .jsp is interpreted and processed by the JSP engine. The JSP engine needs a java compiler in order to process .jsp pages. If the web server and JSP/Servlet engine are external to each other, they must be able to communicate with each other.

It is important to note that many of the concepts discussed in this document including properties file, WEB-INF directory, TLD, web.xml file, tag-lib uri and others are not specific to JSheet and if any of these concepts are covered in this document, it will be only briefly; if at all. There are several books that contain detailed information on JSP applications in general.

For information about your JSP/Servlet environment, consult your system administrator.

JSheet JSP Custom Tags

Chapter 1

Requirements

The requirements to use JSheet JSP Custom tags can be divided into two distinct sections; server-side requirements and client-side requirements.

Server-side Requirements

The Java Server Pages environment itself is very flexible as is the JSheet environment. The components below could be run on a single machine or even multiple machines. It is possible to use the JSheet JSP Custom Tags in a load balanced environment as well.

Due to this flexibility, it is not feasible to specify hardware requirements. Each component will have it's individual hardware requirements. Suffice it to say, the machine(s) should meet or exceed the hardware requirements for the most demanding component.

The following server-side components are required for the JSheet JSP Custom Tags:

- JSheet Server 1.0.7 or higher
- A Web Server
- JSP/Servlet engine that supports JSP1.1 and servlets 2.2 such as Tomcat, JRun or Web Logic
- Java 1.2.2 or higher

As mentioned before, some JSP/Servlet engines have a built in Web Server in which case a separate Web Server will not be required.

If your company will utilize a firewall between JSheet Server and the JSP/Servlet engine, certain ports will be required to be opened.

Component	Connection Type	Inbound Port	Outbound Port Range
JSheet Server	TCP/IP	5000	1024 - 65535
JSheet Server/CORBA	Bi-directional IIOP	1572	1024 - 65535

The inbound ports listed above are configurable in the JSServer Preferences. The TCP/IP to port 5000 by default is a straight TCP/IP connection to retrieve the IIOP connection port. It can be configured to allow for use as an URL or file. See the JSServer documentation for more specific on configuring these settings. See the "JSheet JSP Connect Custom Tag" section in Chapter 3 for information on how to pass the settings to JSClient.

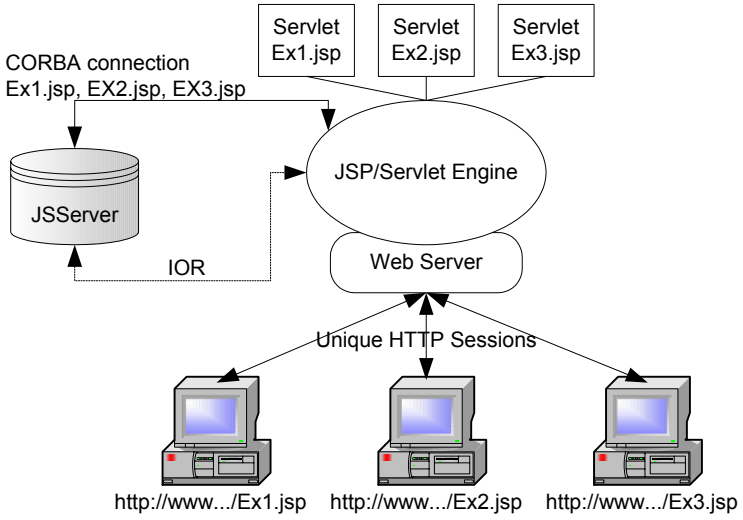
Client-side Requirements

The client-side is the side that is making the JSP page request to the web server. There are no minimum hardware requirements for this side as long as the machine meets or exceeds the minimum hardware requirements for the browser specified below.

- Microsoft Internet Explorer 5.0 or higher

JSheet JSP Custom Tags Component Overview

Basic JSheet JSP Custom Tags



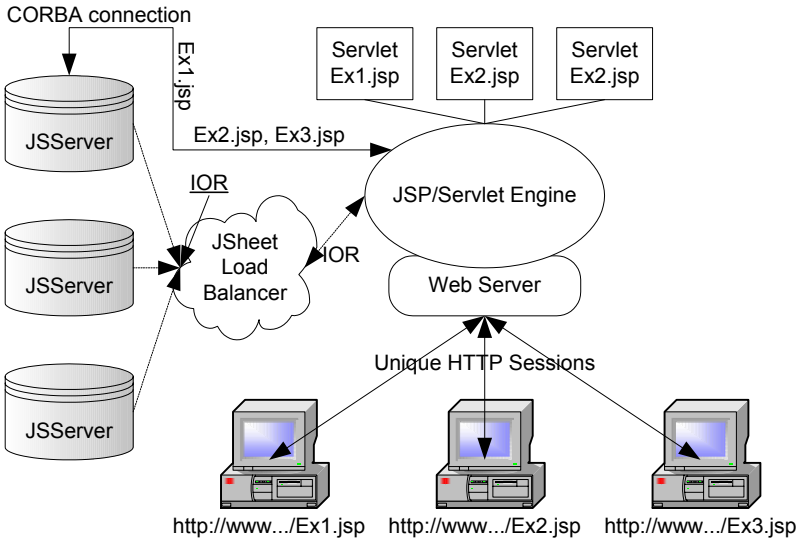
Prior to making any references to JSServer's CORBA objects, JSClient retrieves the JSServer IOR. By default this is done with a straight TCP/IP socket connection to JSServer over port 5000.

It is important to note that the connection to JSServer is made via custom tag generated Servlets in the JSP/Servlet. Therefore, there will not be a unique CORBA connection for each JSP client, and multiple client requests may use the same connection.

The CORBA connection duration is dependent upon how your JSP/Servlet engine manages the duration of the HTTP Session. In the reference implementation of JSP/Servlet engines (Tomcat) once a CORBA connection is made by the JSP/Servlet engine it will be re-used until all HTTP Sessions no longer exist.

As with any servlet & CORBA implementation, settings in the JSP/Servlet engine can affect the CORBA connection duration. For example, if the JSP/Servlet engine is set to cache HTTP Sessions, the CORBA connection may never be destroyed. Setting the HTTP Session time out value too high or too low can impact the CORBA Connection as well. The CORBA connection duration may also be impacted by setting the time out value in the JSheet JSP Connect Custom tag as it determines the HTTP Session duration.

JSheet JSP Custom Tags and the JSheet Load Balancer



Prior to making any references to JSServer's CORBA objects, JSClient retrieves the JSServer IOR. By default this is done with a straight TCP/IP socket connection to JSServer over port 5000. When using the JSheet Load Balancer, each JSServer is launched making it's IOR known to the JSheet Load Balancer. The JSheet Load Balancer then distributes the IOR of the next JSServer in a "round-robin" fashion for each connection request. By default, the Load Balancer retrieves the IOR's via a TCP/IP socket on port 5001.

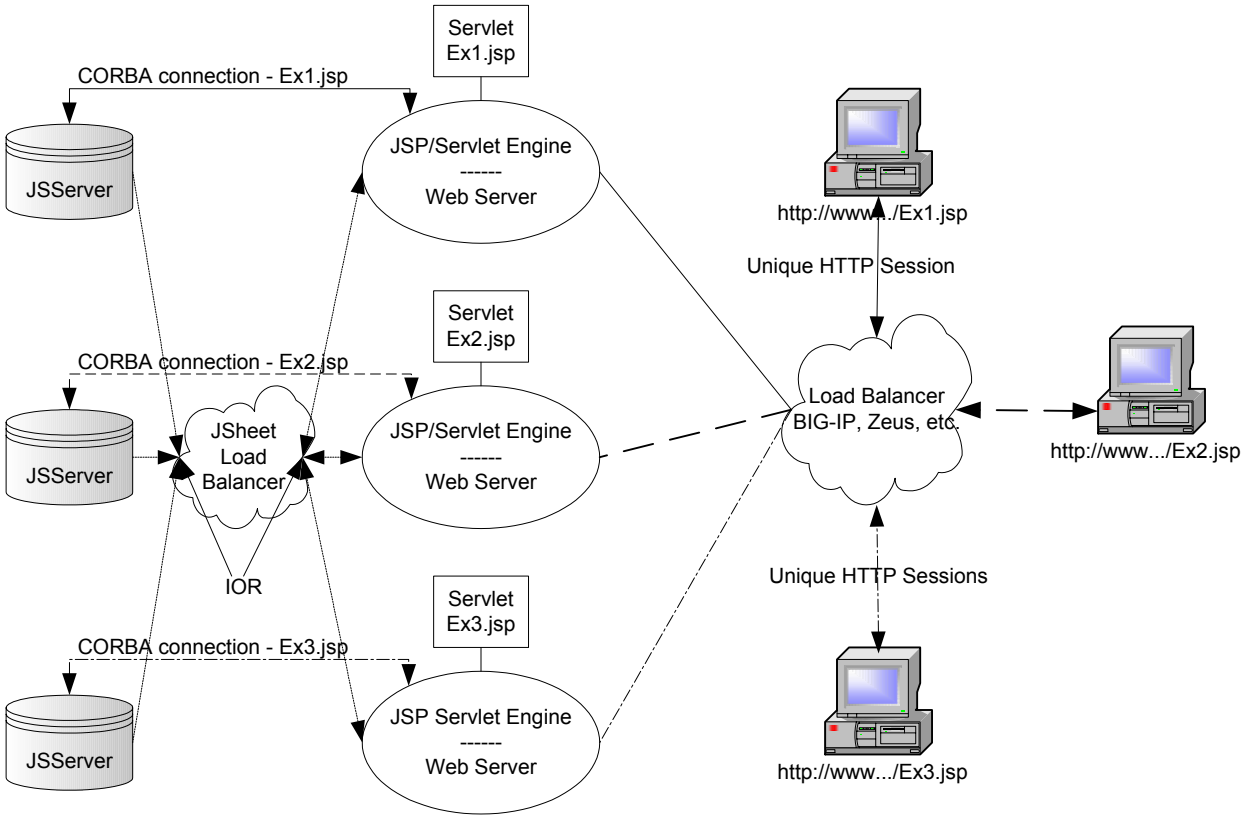
It is important to note that the connection to JSServer is made via custom tag generated Servlets in the JSP/Servlet. Therefore, there will not be a unique CORBA connection for each JSP client, and multiple client requests may use the same connection.

A connection to a different JSServer will not be established until the existing CORBA connection is destroyed, at which time the JSheet Load Balancer will issue the CORBA IOR of the "next" JSServer. This setup creates a psuedo "redundant server" situation ensuring that a connection to a JSServer can be made as long as there is at least one JSServer running and reachable. This should not to be confused with a "fail-over" configuration as one cannot specify that a JSServer be the primary, secondary or tertiary JSServer.

The CORBA connection duration is dependent upon how your JSP/Servlet engine manages the duration of the HTTP Session. In the reference implementation of JSP/Servlet engines (Tomcat) once a CORBA connection is made by the JSP/Servlet engine it will be re-used until all HTTP Sessions no longer exist.

As with any servlet & CORBA implementation, settings in the JSP/Servlet engine can affect the CORBA connection duration. For example, if the JSP/Servlet engine is set to cache HTTP Sessions, the CORBA connection may never be destroyed. Setting the HTTP Session time out value too high or too low can impact the CORBA Connection as well. The CORBA connection duration may also be impacted by setting the time out value in the JSheet JSP Connect Custom tag as it determines the HTTP Session duration.

JSheet JSP Custom Tags, JSheet Load Balancer in a Load Balanced Environment



Prior to making any references to JSServer's CORBA objects, JSClient retrieves the JSServer IOR. By default this is done with a straight TCP/IP socket connection to JSServer over port 5000. When using the JSheet Load Balancer, each JSServer is launched making it's IOR known to the JSheet Load Balancer. The JSheet Load Balancer then distributes the IOR of the next JSServer in a "round-robin" fashion for each connection request. By default, the Load Balancer retrieves the IOR's via a TCP/IP socket on port 5001.

It is important to note that the connection to JSServer is made via custom tag generated Servlets in the JSP/Servlet. Therefore, there will not be a unique CORBA connection for each JSP client, and multiple client requests may use the same connection.

Using a Load Balancer or Multiplexer to distribute HTTP traffic to a server farm in conjunction with the JSheet Load Balancer, causes each JSP/Servlet engine to request a JSServer IOR from the JSheet Load Balancer. A connection to a different JSServer will not be established until the existing CORBA connection for that JSP/Servlet engine is destroyed, at which time the JSheet Load Balancer will issue the CORBA IOR of the "next" JSServer if another connection to JSServer is required.

The CORBA connection duration is dependent upon how your JSP/Servlet engine manages the duration of the HTTP Session. In the reference implementation of JSP/Servlet engines (Tomcat) once a CORBA connection is made by the JSP/Servlet engine it will be re-used until all HTTP Sessions no longer exist.

As with any servlet & CORBA implementation, settings in the JSP/Servlet engine can affect the CORBA connection duration. For example, if the JSP/Servlet engine is set to cache HTTP Sessions, the CORBA connection may never be destroyed. Setting the HTTP Session time out value too high or too low can impact the CORBA

Connection as well. The CORBA connection duration may also be impacted by setting the time out value in the JSheet JSP Connect Custom tag as it determines the HTTP Session duration.

Deployment

The JSheet JSP Custom Tags are distributed as a deployable standard web archive file (.war) named JSheetCustomTags.war.

Deploying the JSheetCustomTag.war file itself depends upon the JSP/Servlet engine you have chosen.

Below is a minimalistic overview of deploying the JSheetCustomTags.war file on the Tomcat and JRun JSP/Servlet engines.

- Tomcat: Place the JSheetCustomTag.war file in the "webapps" directory within the Tomcat directory. Restarting Tomcat will cause Tomcat to find the JSheetCustomTag.war file and automatically create the directory structure and place the files correctly within it.

- JRun: In the JRun Management Console, you can run the "Edit/Create/Deploy and Remove Applications" wizard to deploy the JSheetCustomTag.war file.

The above examples are by no means indicative of the only JSP/Servlet engine you can deploy JSheet JSP Custom Tags on, nor does it give complete instructions.

How you deploy them in your environment will be specific to your JSP/Servlet engine and may be more complex if you are deploying or integrating them into an existing application rather than creating an application from the ground up. Refer to the documentation that came with your JSP/Servlet engine for specific information on deploying "web archive files".

The URL's below contain information that may help increase your understanding of JSP Custom Tags in general and how to work with them.

http://java.sun.com/products/jsp/taglibraries.html

http://java.sun.com/webservices/docs/1.0/tutorial/index.html

Chapter 2

Getting Started

In a JSP page, a custom tag is prefixed with a unique name that distinguishes the tag as a custom tag. The Taglib Directive is used to assign that unique prefix name and to point to what is called the Tag Library Descriptor. The Tag Library Descriptor describes the semantics of the custom tags. Below is an example of a Taglib Directive:

```
<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>
```

In the above example we create a prefix by using the Taglib Directive's prefix attribute. The prefix attribute is set to **"jsheet"**. The location of the Tag Library Descriptor is established through the **uri** attribute. In this case we set it to **"/JSheetExamples"**. The **uri** attribute value must match the value used in the **<taglib-uri>** of the **web.xml** file.

Now that the prefix and Tag Library Descriptor have been established, we can use the custom tags by prefixing **"jsheet:"** to them. Below is an example of how to use the **button** custom tag with the defined ***jsheet*** prefix:

```
<jsheet:button/>
```

Naturally, since the Taglib Directive sets the structure for the remaining page, the Taglib Directive must appear above all custom tags in the JSP page.

Example

```
<html>
<body>

<!-- This is the taglib directive.-->
<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<!--Since the taglib directive sets the prefix to 'jsheet' all custom tags will have
'jsheet:' prefixed to them. -->
<jsheet:connect
       sheet="0"
       name="JSheetExamples"
       openbookname="/JSheetExamples/examples.jss">
<jsheet:form>
<jsheet:button/>
</jsheet:form>
</jsheet:connect>
</body>
</html>
```

File Locations

Since JSheet Custom Tag classes are in the **JSheet.jar** and the **JSheetTagLibrary Server.jar**, the JSP/Servlet engine needs to be able to locate these two archives. This is generally accomplished by putting the **JSheet.jar** and the **JSheetTagLibraryServer.jar** in the classpath. The default location for this is ***ApplicationName*/WEB-INF/lib/**.

If using the thin client proxy in **interactive mode**, the **JSheetTagLibraryClient.jar** must be placed in the same directory as the serving **jsp** page. The default location for this is the top level directory of the web application.

If using the full JSClient applet in **interactive mode**, the **JSheet.jar** file must be in a location available to the applet. The default location for this is the top level directory of the web application.

The **web.xml** file describes the mapping between the taglib **uri** and the location of the Tag Library Descriptor. The default location for this file is ***ApplicationName*/WEB-INF/.**

The **Tag Library Descriptor** (TLD) is an XML document that maps action tags to tag handler classes. The **web.xml** elements **`<taglib-uri>`** or **`<tablib-location>`** may specify where the TLD can be found. The default location for this file is ***ApplicationName*/WEB-INF/**{taglibrary**}.tld.**

Error Handling

The first time a JSP page is loaded, it is compiled. Compilation errors (if any) will be displayed by the JSP/Servlet engine.

Validation errors are returned at runtime. Although JSP processing does not attempt to check for every possible error condition, common errors will be caught and meaningful error messages will be displayed on the generated web page. If a JSP page contains an invalid value that is not explicitly checked by JSP processing, JSheet will throw a standard Java exception to indicate the error. The JSP tag handler will then capture that exception and display the error message as part of its generated web page.

Custom Tag Attributes

Most JSP custom tags have attributes. For each JSP custom tag that has attribute(s), some may be required. For required attributes, you must specify a value. Use lowercase text for all tag names and attributes.

To assign a value to an attribute, you can:

- Hardcode the value directly in the JSP page.
- Include attribute value pairs in the URL used to load the **.jsp** file. You can use this technique for any attribute except the **name** attribute.

Most of the custom tags have a **name** attribute. The tag **name** attribute will be assigned a default value if you do not explicitly give it a value. You can use this tag name when you assign tag attribute values with an URL to associate the attribute values with the named tag. The **name** attribute can also be used to write JavaScript code that accesses the specified tag. The JavaScript code may then read the tag's attribute values or it may update the tag's attribute values.

If specifying an attribute value as part of a URL, you may optionally prefix the attribute name with the tag **name.** If the tag **name** qualifier is used, then the attribute assignment applies only to the named tag. If the tag **name** prefix is omitted, then the attribute assignment applies to all tags that use the specified attribute. Here is an example URL:

```
http://mydomain.com/test.jsp?myButtonTag.action=save&sheet=0
```

The above URL loads the JSP page named **test.jsp** from the site **mydomain.com**. The JSP page **test.jsp** contains a button custom tag with the **name** attribute set to **myButtonTag**. Below is how the tag should appear in **test.jsp**:

```
<jsheet:button
        name="myButtonTag"/>
```

The button custom tag also has an **action** attribute. Since the **action** attribute is not explicitly set in the above example, it is assigned is assigned the value **save**. This was specified in the URL that loaded the JSP page **test.jsp**, with the following fragment:

```
myButtonTag.action=save
```

In addition, for each tag in **test.jsp** that contains a **sheet** attribute, the **sheet** attribute is assigned the value *0*. This was set in the URL with the following fragment:

```
sheet=0
```

The following search order is used to determine the value of an attribute:

- First, if the custom tag's name is specified in the URL with an attribute and value, then that specific tag receives that specific attribute value. So if the URL contains **myButtonTag.action=submit**, the custom tag named **myButtonTag** will be assigned the value **submit** for the **action** attribute. If the tag's **name** attribute value is not unique in the JSP page (i.e., there are more than one button tags are named **myButtonTag**), the assignment(s) are made to all tags with the same name.
- Second, if an attribute value is defined in the URL without a tag name that attribute value is used for all custom tags on the page with that attribute. So if the URL contains **sheet=0**, all tags with the **sheet** attribute will be assigned a value of **0**. The assignment is not applied to custom tags that do not have an associated **sheet** attribute.
- Third, if the JSP page contains a custom tag block that assigns an attribute a value, then the value contained on the JSP page is used.
- Finally, the default value as specified in the custom tag's documentation is used.

Defining Pass-Through Attributes

Many of the JSP custom tags allow pass-through attributes. The pass-through attributes are expected to work as specified by the generated tag's HTML 4.0 tag specification. However, browsers typically implement only a subset of the HTML specification. To ensure proper performance when using custom tags, you must check the specifications for your particular browser(s) before using the pass-through attributes.

Accessing the JSheet client within the JSP page

By using an instance of JSheet's **JSClient** class, you may write code that interacts with the JSheet server during the processing of the JSP custom tags. To access an instance of **JSClient**, include this line of code in a JSP page:

```
JSClient jsClient = (JSClient) session.getAttribute("jsclient");
```

You may then use the **jsClient** object to call any of the API methods in the **JSClient** class.

Example

```
<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<%@ page import="com.iisc.jwc.jsheet.*" %>
<%@ page import="com.iisc.jwc.jsheet.db.JSDatabase" %>
<%@ page import="com.iisc.jwc.jsheet.db.JSDbCursor" %>

<html>
<head>
<title>Demo Database</title>
</head>

<jsheet:connect
       sheet="0"
       name="JSheetExamples"
       openbookname="/JSPCustomTagExamples/Database.jss">
<%
JSClient jsclient = (JSClient)session.getAttribute("jsclient");
try {
       JSDatabase jsdb = jsclient.dbConnect("Text","","");

       JSDbCursor jsdbc;

       jsdbc = jsdb.runQuery("Select * from EMP",false);

       jsdbc.closeCursor();
}
catch (Exception e) {
       out.println("<p>Exception 1: " + e.toString() + "</p>");
       e.printStackTrace();
}
%>
```

```
</jsheet:connect>
</body>
</html>
```

IMPORTANT: Since the custom tag handlers use that same instance of JSClient, do not change the state of the **jsClient** object. For example, closing the workbook or disconnecting from the server will all cause problems with JSP tag processing.

Working with the JSheetTagLibrary.properties File

You can use the **JSheetTagLibrary.properties** file to store various configuration settings. In order for the JSP/Servlet engine to find the **JSheetTagLibrary.properties** file, it should typically be located in the WEB-INF/classes directory. The **JSheetTagLibrary.properties** is read only once when the JSP/Servlet engine is first started. Any changes to the properties file after the engine has been started will require the JSP/Servlet engine to be restarted. For more general information on the properties file, please see the documentation that came with your JSP/Servlet engine.

In this file, many of the properties are prefixed with a **name**. The **name** is provided by the **connect** and the **database** tag's **name** attribute. For example, if the **connect** tag's **name** attribute is **user1**, then **user1.host** is searched for in the properties file to find the value of the **host** attribute. Below is an example of how the **connect** tag will appear in order to use the **JSheetTagLibrary.properties** file attributes' settings:

```
<jsheet:connect
       name="user1">
```

Since the **name** attribute is specified, the **JSheetTagLibrary.properties** file will be searched for all attributes prefixed with value of the **name** attribute. In the above example the **JSheetTagLibrary.properties** file will search for attributes that are prefixed with **user1**. The following **JSheetTagLibrary.properties** file shows an example of property settings.

```
user1.host=demo.jsheet.com
user1.user=demo
user1.password=demo
user1.sessiontimeout=2
user1.port=5001
user1.dbname=myDatabase
user1.dbuser=guest
user1.dbpassword=guest
chartservlet=ChartTagServlet
jssupportpath=/jssupport
```

The following table provides information about each property setting:

Attribute	Description
***name*.host**	The host name for the machine where JSheet server is running.
***name*.user**	The user name used for connecting with the JSheet server host.
***name*.password**	The password used for connecting with the JSheet server host.
***name*.jsclientproxyclient**	The servlet that the JSClientProxy applet needs to communicate with during interactive mode. This servlet is located in the JSheetTagLibraryServer.jar in the com.iisc.jsheet.taglib.servlets.JSClientProxy package.
***name*.appletpoll**	How often, in seconds, the proxy applet automatically queries the proxy servlet for events in interactive mode.
***name*.applet.name**	The name attribute of the JSClient applet. This is used if the full JSClient applet is desired for interactive mode.
***name*.applet.codebase**	The URL or URI of the directory of the JSClient applet.
***name*.sessiontimeout**	The timeout period, in minutes, for a session.
***name*.port**	The port number for the machine where JSheet Server is running.
***name*.dbname**	The name of the database to access.

***name.*dbuser**	The user name needed to access the specified database.
***name.*dbpassword**	The password needed to access the specified database.
jssupportpath	The path to the JSheet support files (i.e., JavaScript, html, images). By default this is under a directory called jssupport.
chartservlet	The servlet with which the <chart> tag communicates. This servlet is located in the JSheetTagLibraryServer.jar the com.iisc.jsheet.taglib.servlets.ChartTagServlet package.

When you explicitly specify property-related attributes within a JSheet custom tag block, the values you specify override values in the **JSheetTagLibrary.properties** file.

Chapter 3: The JSheet Custom Tags

button

Use this tag to create a button on a form.

Syntax

<jsheet:button
[**name=**"*string*"]
[{**type=**"button"} | {**type=**"image" **src=**"*string*"}]
[**action=**{"reset" | "submit" | "save"}]
[**click=**"*string*"]
[**change=**"*string*"]
[classname="*string*"]
[*pass through attributes*]**/>**

This tag must be nested inside a **<form> </form>** custom tag pair, and the **<form> </form>** custom tag pair must be nested inside a **<connect> </connect>** custom tag pair.

In **batch** mode if the button action is "*submit*", the form is submitted when the user presses the button and the values are updated in the workbook. If the button action is "*save*", the form is submitted when the user presses the button, the values are updated in the workbook, and the workbook is saved. If the button action is "*reset*", the form controls are all reset to the value they originally contained.

In **interactive** mode if the button action is "*submit*", an error will be displayed. If the button action is "*save*", then pressing the button causes the workbook to be saved. The action "*reset*" is not available in **interactive** mode.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
name	No	Auto-generated as follows: if **<submit>** is the second JSP custom tag on the current page, the generated HTML **<input>** element is named **tag2**.	The name of the generated input control.
type	No	button	The type of control to display. Valid values are **image** and **button**.
src	No	None	The file name of the image to use for the button. This is only valid if the **type** attribute is set to **image**.
action	No	submit	The action to perform when the button is clicked. Possible values are **submit**, **reset** and **save**. The values **submit** and **reset** are only available in batch mode.
click	No	None	The standard HTML attribute, **onClick**, is not available for use since it interferes with dynamic sheet updates. The attribute **click** replaces **onClick**. This attribute provides the same functionality but it is used differently. When in **interactive** mode, the attribute value for **click** *must* be the

			name of a JavaScript function that returns a **true** or **false** value. If the return value of the function is **true,** the spreadsheet will be updated with the HTML control value. If the return value of the function is false, the spreadsheet will not be updated with the HTML control value. If the mode is **batch**, the attribute value defined will be placed into the standard HTML attribute **onClick**.
classname	No	None	This is to be used instead of the class passthrough attribute.

Pass-Through Attributes

The following optional attributes that are passed through the JSP page generation phase without undergoing any changes. The attributes are not processed by the JSP page generation phase, but added to the generated **<input type="submit">** tag exactly as they are entered. The attributes are expected to work as specified by the HTML 4.0 **<input type="submit">** tag specification

accept, accesskey, align, alt, class (use classname instead), dir, disabled (use disabled="true"), id, ismap, lang, onblur, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, size, style, tabindex, title, usemap, value

Example

```
<html>
<head>
<title>Button</title>
</head>
<body>
<h1>Button</h1>
<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       sheet="0"
       name="JSheetExamples"
       shared="true"
       openbookname="/JSPCustomTagExamples/examples.jss"
       timeout="2">

<jsheet:form>

<jsheet:button
       action="submit"/>

</jsheet:form>
</jsheet:connect>

</body>
</html>
```

chart

Use this tag to generate an **<img>** tag with its **src** attribute pointing to the **chartservlet** value. **chartservlet** contains the servlet that generates the chart. The result is a **.gif** image of a chart. The chart is specified by the **chartname** attribute or a combination of the **charttemplate**, **range**, **height**, and **width** attributes.

Syntax

<jsheet:chart
[**name=**"*string*"]
[**chartservlet=**"*string*"]
{{**chartname=**"*string*"[**height=**"*nonnegative_int*" **width=**"*nonnegative_int*"]}
|{**charttemplate=**"*string*" **range=**"*string*" **height=**"*nonnegative_int*" **width=**"*nonnegative_int*"}}
[**sheet=**{"*string*" | "*nonnegative_int*"}]
[**click=**"*string*"]
[**autorefresh**="*boolean*"]
[**classname**="*string*"]
[*pass through attributes*]**/>**

This tag must be nested inside a **<connect> </connect>** custom tag pair.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
name	No	Auto-generated as follows: if **<chart>** is the second JSP custom tag on the current page, the generated **HTML <img>** element is named **tag2**.	The name of the generated **HTML <img>** element.
chartservlet	No	None	The servlet that generates the chart. If this attribute is defined here, this value overrides the **chartservlet** setting in the properties file.
chartname	Only if the **charttemplate** and **range** attributes are not specified.	None	The name of an existing chart on the current book.
height	Only if the **charttemplate** and **range** attributes are specified, in which case, you must also specify the **width** attribute.	None	A non-negative integer value that represents height in pixels of the chart image. If you specify the **height** attribute in conjunction with the **chartname** attribute, the chart image might be skewed.
width	Only if the **charttem-plate** and **range** attributes are specified, in which case, you must also specify the **height** attribute.	None	A non-negative integer value that represents width in pixels of the chart image. If you specify the **width** attribute in conjunction with the **chartname** attribute, the chart image might be skewed.
charttemplate	Only if the **chartname** attribute is not specified. If you specify this	None	The path and name of the template file to use when creating the chart, such as an **.xml** file. This must be a location relative to

	attribute, you must also specify the **height**, **width**, and **range** attributes.		the JSServer workbook path.
range	Only if the **charttemplate** attribute is used or if the mode is equal to **interactive**.	No	The range that contains the data to chart. The range's value should be a cell range (i.e., B2..D4) or a valid range name.
sheet	No	Setting from **<connect>**	The sheet associated with the **range**'s attribute value. This value can be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
autorefresh	No	true	This attribute is only valid in interactive mode. If true, then the control is updated whenever the control's associated cell is changed in the applet. If false, then the control is not updated whenever the control's associated cell is changed in the applet.
click	No	None	The standard HTML attribute, **onClick**, is not available for use since it interferes with dynamic sheet updates. The attribute **click** replaces **onClick**. This attribute provides the same functionality but is used differently. When in **interactive** mode, the attribute value for **click** *must* be the name of a JavaScript function that returns a **true** or **false** value. If the return value of the function is **true,** the spreadsheet will be updated with the HTML control value. If the return value of the function is false, the spreadsheet will not be updated with the HTML control value. If the mode is **batch**, the attribute value defined will be placed into the standard HTML attribute **onClick**.
classname	No	None	This is to be used instead of the class passthrough attribute.

Pass-Through Attributes

The following optional attributes are passed through the JSP page generation phase without undergoing any changes. The attributes are not processed by the JSP page generation phase, but added to the generated **<img>** tag exactly as they are entered. The attributes are expected to work as specified by the HTML 4.0 **<img>** tag specification:

align, alt, border, class (use classname instead), dir, hspace, id, lang, longdesc, ondblclick, onkeydown, onkeypress, onkeyup,

onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, style, title, vspace.

Example

```
<html>
<head>
<title>Chart</title>
</head>
<body>
<h1>Chart</h1>
<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       sheet="0"
       name="JSheetExamples"
       shared="true"
       openbookname="/JSPCustomTagExamples/examples.jss"
       timeout="2">

<jsheet:chart
       chartservlet="ChartTagServlet"
       chartname="myfirstchart"
       sheet="0" />
</jsheet:connect>

</body>
</html>
```

checkbox

Use this tag to create a check box input control.

Syntax

<jsheet:checkbox
  [**name=**"*string*"]
  [**sheet=**{"*string*" | "*non-negative_int*"}]
  cell="*string*"
  [**autorefresh=**"*boolean*"]
  [**updatesheet=**"*boolean*"]
  [{**labelcell=**"*string*" [**labelsheet=**{"*string*" | "*non-negative_int*"}]}
    | {**labeltext=**"*string*"}]
  [**truevalue=**{"*string*" | "*int*"}]
  [**falsevalue=**{"*string*" | "*int*"}]
  [**click=**"*string*"]
  [**change=**"*string*"]
  [**classname=**"*string*"]
  [*pass through attributes*]**/>**

If the value in the check box control's associated cell is not equal to the **truevalue** or **falsevalue** attributes, the checkbox defaults to an unchecked state.

This tag must be nested inside a **<form> </form>** custom tag pair, and the **<form> </form>** custom tag pair must be nested inside a **<connect> </connect>** custom tag pair.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
name	No	Auto-generated as follows: if **<checkbox>** is the second JSP custom tag on the current page, the generated **HTML <input>** element is named **tag2**.	The prefix of the generated **<input>** control.
sheet	No	The value from the **sheet** attribute of the **<connect>** tag.	The sheet that contains the **cell** attribute's value. This value can be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
cell	Yes	None	The target cell to contain the selected checkbox value. This value should be a cell (i.e., A1 or R1C1) or a range name that references one cell. The sheet that contains the cell or range should be specified by the **sheet** attribute.
autorefresh	No	True	This attribute is only valid in inter-active mode. If true, then the control is updated whenever the control's associated cell is changed in the applet. If false, then the control is not updated whenever the control's associated cell is changed in the applet.

labelcell	No	The value from the **cell** attribute.	The cell that contains the label to be displayed to the right of the checkbox. This value should be a cell (i.e., A1 or R1C1) or a range name that references one cell. The sheet that contains the cell or range should be specified by **labelsheet** attribute. If you include this attribute, do not include the **labeltext** attribute.
labelsheet	No	The value from the **sheet** attribute.	The sheet that contains the **labelcell** attribute's value. This value can be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
labeltext	No	None	The label to display instead of the label specified by the **labelcell** attribute. If you include this attribute, do not include the **labelcell** attribute.
truevalue	No	1	The value used to indicate that the checkbox is checked.
falsevalue	No	0	The value used to indicate that the checkbox is not checked.
updatesheet	No	True	This attribute is only valid in **inter-active** mode. If this value is **true**, then the control's associated applet cell is updated whenever the control is changed. If this value is **false**, then the control's associated applet cell is not updated whenever the control is changed.
click, change	No	None	The standard HTML attributes, **onClick** and **onChange,** are not available for use because they interfere with dynamic sheet updates. The attributes **click** and **change** replace **onClick** and **onChange**. These attributes provide the same functionality but are used differently. When in **interactive** mode, the attribute value for **click** or **change** *must* be the name of a JavaScript function that will return a **true** or **false** value. If the return value of the function is **true,** the spreadsheet will be updated with the HTML control value. If the return value of the function is false, the spreadsheet will not be updated with the HTML control value. If the mode is **batch**, the function will be passed-through to the standard HTML attributes **onClick** and **onChange**.

classname	No	None	This is to be used instead of the class passthrough attribute.

Pass-Through Attributes

The following optional attributes are passed through the JSP page generation phase without undergoing any changes. The attributes are not processed by the JSP page generation phase, but added to the generated **<input type="checkbox">** tag exactly as they are entered:

accept, accesskey, align, class (use classname instead), dir, disabled (use disabled=true), id, lang, onblur, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, onselect, size, style, tabindex, title

Example

```
<html>
<head>
<title>CheckBox</title>
</head>
<body>
<h1>CheckBox</h1>
<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       sheet="0"
       name="JSheetExamples"
       shared="true"
       openbookname="/JSPCustomTagExamples/examples.jss"
       timeout="2">

<jsheet:form>

<jsheet:checkbox
       sheet="0"
       cell="A5"
       labelcell="A4"
       labelsheet="0"
       labeltext="selectionone"
       truevalue="1"/>

</jsheet:form>
</jsheet:connect>

</body>
</html>
```

combobox

Use this tag to create a combo box control on a form.

Syntax

<jsheet:combobox
  [**name=**"*string*"]
  [**sheet=**{"*string*" | "*non-negative int*"}]
  cell="*string*"
  [**autorefresh**="*boolean*"]
  [**updatesheet**="*boolean*"]
  {{**valuerange=**"*string*" [**valuesheet=**{"*string*" | "*non-negative int*"}]}
    | {**valuelist=**"*string*"}}
  [{**labelrange=**"*string*" [**labelsheet=**{"*string*" | "*non-negative int*"}]}
    | {**labellist=**"*string*"}]
  [**click=**"*string*"]
  [**change=**"*string*"]
  [classname="*string*"]
  [*pass through attributes*]**/>**

This tag must be nested inside a **<form> </form>** custom tag pair, and the **<form> </form>** custom tag pair must be nested inside a **<connect> </connect>** custom tag pair.

valuerange and **valuelist** are mutually exclusive attributes. **labelrange** and **labellist** are also mutually exclusive attributes. The number of values in the **valuerange/valuelist** attribute takes precedence over the number of values in the **labelrange/labellist** attribute. Thus, the number of displayed labels is determined by the number of **valuerange/valuelist** values.

If the **valuerange/valuelist** attribute contains more values than the **labelrange/labellist** attribute, then the extra values in the **valuerange/valuelist** attribute are used as "fill-ins" for the missing labels from the list of labels in the **labelrange/labellist** attribute. For example, if there are six values in the **valuerange/valuelist** attribute and only four values in the **labelrange/labellist** attribute, then the last two values in the **valuerange/valuelist** attribute are used as the last two labels.

If the **valuerange/valuelist** attribute contains less values than the **labelrange/labellist** attribute, then the extra values in the **labelrange/labellist** attribute are ignored. For example, if there are ten values in the **valuerange/valuelist** attribute and eleven values in the **labelrange/labellist** attribute, then the last value in the **labelrange/labellist** attribute is not displayed.

If **valuerange|valuelist** is specified and **labelrange|labellist** is not specified the items displayed in the combobox and the value written to the JSheet worksheet will be the same. If both the **valuerange|valuelist** and the **labelrange|labelist** attributes are specified, the items displayed in the combobox will be taken from **labelrange|labellist**. However, the actual value written the JSheet worksheet will be taken from the corresponding **valuerange|valuelist** attribute.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
name	No	Auto-generated as follows: if **<combobox>** is the second JSP custom tag on the current page, the generated HTML **<select>** element is named **tag2**.	The name of the generated **<input>** control.
sheet	No	The value from the **sheet** attribute of the **<connect>**	The sheet that contains the **cell** attribute's value. This value can be the index or the name of the sheet.

		tag.	Sheet indexes start at 0 for the first sheet.
cell	Yes	None	The target cell to contain the selected combobox values. This value should be a cell (i.e., A1 or R1C1) or a range name that references one cell.
autorefresh	No	true	This attribute is only valid in **inter-active** mode. If this value is **true**, then the control is updated whenever the control's associated cell is changed in the applet. If this value is **false**, then the control is not updated whenever the control's associated cell is changed in the applet.
valuerange	Only if the **valuesheet** attribute is specified and **valuelist** is not specified.	None	The range that contains the combobox values. This value should be a cell range (e.g., A1, R1C1, or A1..D4) or a range name. The range size should match the range size specified for the associated **labelrange** attribute. If the **labelrange** attribute is not defined, the **valuerange** values are displayed as the combobox items. It is the **valuerange/valuelist** value that is written to the cell. The sheet that contains this range is specified by the **valuesheet** attribute. If **valuesheet** is not defined, the **sheet** attribute value is used. If the **sheet** attribute is not defined, the value is taken from the connect tag's **sheet** attribute.
valuesheet	Only if the **valuerange** attribute is specified.	The value of the **sheet** attribute.	The sheet that contains the **valuerange**'s attribute value. This value can be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
valuelist	Only if the **valuerange** and **valuesheet** attributes are not specified.	None	The list of combobox values. Use a vertical bar (|) delimited list. For example: "house|car|business" The number of values specified here should match the number of values specified for the associated **labellist** attribute. If the **labellist** attribute is not defined, the **valuelist** values are displayed as the combobox items. The **valuelist** list item selected will be written to the JSheet worksheet.
labelrange	Only if the **labelsheet**	The value of the **valuerange** attribute.	The range that contains the labels to be displayed as the combobox items.

	attribute is specified.		This value should be a cell range (e.g., A1, R1C1, or A1..D4) or a range name. The range size should match the range size specified for the associated **valuerange** attribute. The sheet that contains this range is specified by the **labelsheet** attribute. If **labelsheet** is not defined, the **sheet** attribute value is used. If the **sheet** attribute is not defined, the value is taken from the connect tag's **sheet** attribute. The **labelrange** value is simply the item displayed in the combox. The actual value written to the JSheet worksheet will be take from either **valuerange** or **valuelist**.
labelsheet	Only if the **labelrange** attribute is specified.	The value of the **valuesheet** attribute.	The sheet that contains the **label-range**'s attribute value. This value can be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
labellist	Only if the **labelrange** and **labelsheet** attributes are not specified.	None	A list of labels to be displayed as the combobox items. Use a vertical bar (|) delimited list. For example: "house|car|business" The number of values specified here should match the number of values specified for the associated **valuelist** attribute. The **labellist** value is simply the item displayed in the combox. The actual value written to the JSheet worksheet will be take from either **valuerange or valuelist**.
updatesheet	No	true	This attribute is only valid in **inter-active** mode. If this value is **true**, then the control's associated applet cell is updated whenever the control is changed. If this value is **false**, then the control's associated applet cell is not updated whenever the control is changed.
click, change	No	None	The standard HTML attributes, **onClick** and **onChange,** are not available for use because they interfere with dynamic sheet updates. The attributes **click** and **change** replace **onClick** and **onChange**. These attributes provide the same functionality but are used differently. When in **interactive** mode, the

			attribute value for **click** or **change** *must* be the name of a JavaScript function that will return a **true** or **false** value. If the return value of the function is **true,** the spreadsheet will be updated with the HTML control value. If the return value of the function is false, the spreadsheet will not be updated with the HTML control value. If the mode is **batch**, the function will be passed-through to the standard HTML attributes **onClick** and **onChange**.
classname	No	None	This is to be used instead of the class passthrough attribute.

Pass-Through Attributes

The following optional attributes are passed through the JSP page generation phase without undergoing any changes. The attributes are not processed by the JSP page generation phase, but are added to the generated **<select>** tag exactly as they are entered. The attributes are expected to work as specified by the HTML 4.0 **<select>** tag specification:

class (use classname instead), dir, disabled (use disabled="true"), id, lang, onblur, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, size, style, tabindex, title

Example

```
<html>
<head>
<title>ComboBox</title>
</head>
<body>
<h1>ComboBox</h1>
<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       sheet="0"
       name="JSheetExamples"
       shared="true"
       openbookname="/JSPCustomTagExamples/examples.jss"
       timeout="2">

<jsheet:form >

<jsheet:combobox
       sheet="0"
       cell="A6"
       valuelist="car|house|business"/>

</jsheet:form>
</jsheet:connect>

</body>
</html>
```

connect

Use the **<connect> </connect>** tag pair to define the connection information for the page. This includes information about which book to open or create and the default sheet.

Syntax

```
<jsheet:connect
  [name="string"]
  [host="string"]
  [user="string"]
  [port="port"]
  [password="string"]
  [sheet={"string" | "nonnegative_int"}]
  [{mode="batch" | "palm"}
    | {mode="interactive" servlet="string" [appletpoll="nonnegative_int"]}
    | {mode="interactive" applet="string"}]
  {{openbookname="string" [shared="boolean"]}
    | {newbookname="non-empty string" [shared="boolean"]}
    | {newbookname="" [shared="false"]}
    | {templatename="string" newbookname="non-empty string" [shared="boolean"]}
    | {templatename="string" [newbookname=""] [shared="false"]}}
  [timeout="nonnegative_int"]
  [javascriptexceptions="boolean"]
  [loadbalance="boolean"]
</jsheet:connect>
```

For each JSP page, **<connect> </connect>** must surround all other JSP custom tags. This pair must not be nested inside any other custom tag pairs. Since interactive mode is dynamic, only one **connect** tag may have it's mode as **interactive** per JSP page.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
applet	No	None	The **name** attribute of the JSheet applet on the HTML page. If the applet attribute is not specified, then the thin applet (with no GUI) is automatically generated. The generated applet works in conjunction with the servlet that is specified in either the servlet attribute or the properties file.
mode	No	batch	If the mode attribute's value is **batch** or is not specified, the JSP page does not generate an applet tag. In batch mode, when data is changed on the server, the HTML page does not dynamically update. Likewise, when data is changed on the HTML page, the server data does not dynamically update. The user's entered values on the HTML page are submitted to the server when the submit button is clicked. Also, when the submit button is clicked, the HTML page is refreshed with data from the

			server. If the mode attribute's value is **batch**, then there should be a **<submit>** custom tag within a **<form></form>** custom tag pair. If the mode attribute's value is **interactive**(i.e., interactive mode), then an HTML <applet> tag is generated. In interactive mode, when data is changed on the server, the HTML page dynamically updates. Likewise, when data is changed on the HTML page, the server data is automatically updated. If the mode attribute's value is **palm**, then the tags will simulate a mode similar to batch mode. However, palm mode has been optimized for a Palm handheld device.
name	No	tag1	Refers to the **<connect>** tag or is used as the prefix value found in the properties file.
host	No	None	The name of the JSheet Server host. If this attribute is defined here, this value overrides the **host** setting in the properties file.
user	No	None	The JSP servlet uses this username when connecting to the JSheet Server host. If this attribute is defined here, this value overrides the **user** setting in the properties file.
password	No	None	The JSP servlet uses this password when connecting to the JSheet Server host. If this attribute is defined here, this value overrides the **password** setting in the properties file.
sheet	No	0	The sheet in the book that you want to be used. This value can be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
openbookname	No*	None	The name of an existing book that you want opened. Only one book can be opened.This will point to a location relative the JSServer path.
shared	No	false	True indicates that other users can open the same book collaboratively. When users make changes to a book, other users can click the submit button to see those changes. If the

			loadbalance attribute is true, this attribute must be set to false.
newbookname	No*	None	The name of the new book to be created. Only one book can be created. To generate a new name rather than identify an existing name, specify the attribute as follows: **newbookname=""**. If a book is opened shared, it must be given a name.
templatename	No*	None	The name of the template file to use when creating a new book.
timeout	No	30	The timeout period in minutes for a session. If this attribute is defined here, this value overrides the **timeout** setting in the properties file.
servlet	No	None	The name of the servlet in the form of a full URL. The servlet acts as the intermediary between the generated applet and the JSheet server. Normally specified in the properties file.
appletpoll	No	10 seconds	How often, in seconds, the proxy applet automatically queries the proxy servlet for events. This is only valid in interactive mode.
loadbalance	No	false	Indicates whether server access is through the LoadBalancer or not. If this is true, the port number must match the LoadBalancer's clientport parameter.
javascriptexceptions	No	true	True indicates that default JavaScript exception handlers should be added to the HTML output.

*NOTE: **One of the three parameters:** *openbookname*, *newbookname*, **or** *templatename* **must be specified.**

Pass-Through Attributes

This tag has no pass-through attributes.

Example

```
<html>
<head>
<title>Connect</title>
</head>
<body>
<h1>Connect</h1>
<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       host="demo.jsheet.com"
       user="demo"
       password="demo"
```

```
        sheet="0"
        shared="true"
        openbookname="/JSPCustomTagExamples/examples.jss"
        timeout="2">
</jsheet:connect>

</body>
</html>
```

database

Use the **<database>** custom tag to establish a connection to an **ODBC** database.

Syntax

<jsheet:database
  [**name=**"*string*"]
  [**dbname=**"*string*"]
  [**dbuser=**"*string*"]
  [**dbpassword=**"*string*"]**>**
</jsheet:database>

This tag pair must be nested inside a **<connect> </connect>** custom tag pair. The **<database> </database>** custom tag pair should surround all JSP custom database tags. The current list of JSP custom database tags is as follows:

<executequery>, **<fetchinto>**

Attributes Processed by JSP

Attribute	Required	Default Value	Description
name	no	None	The name associated with the database tag.
dbname	Yes	None	The name of the ODBC database.
dbpassword	Yes	None	The password for the database connection.
dbuser	Yes	None	The user name for the database connection.

Pass-Through Attributes

This tag has no pass-through attributes.

Example

```
<html>
<head>
<title>Database</title>
</head>
<body>
<h1>Database</h1>

<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       name="JSheetExamples"
       openbookname="/JSPCustomTagExamples/example.jss"
       sheet="0">

<jsheet:database
       dbpassword="demo"
       dbname="myDatabase"
       dbuser="myDatabaseUser">

<jsheet:executequery>
       Select * from myTable
</jsheet:executequery>

</jsheet:database>
```

```
</jsheet:connect>

</body>
</html>
```

date

Use the **<date>** custom tag to display a date input control.

Syntax

<jsheet:date
[**name=**"*string*"]
[**sheet=**{"*string*" | "*nonnegative int*"}]
cell="*string*"
[**showcellentry=**"*boolean*"]
[**click=**"*string*"]
[**change=**"*string*"]
[**classname=**"*string*"]
[*pass through attributes*]**/>**

This tag pair must be nested inside a **<form> </form>** custom tag pair. The **<form> </form>** custom tag pair should be nested inside a **<connect> </connect>** custom tag pair.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
name	no	The default value is auto-generated as follows. If **<date>** is the second JSP custom tag on the current page, then the generated HTML **<input>** element would be named tag2.	Must be a valid JavaScript identifier - each character may be a letter, underscore, dollar sign, or digit, but the first character may not be a digit.
sheet	no	Sheet setting from the **<connect>** tag.	The sheet that contains the data to be written. This can either be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
cell	yes	None	The cell that contains the data to be written. Cell's value should be a reference (A1) or a range name. The specified cell's value is used as the value of the generated text control's value attribute.
showcellentry	no	false	If false the cell's display is displayed in the control. For example, if the cell contained "*=year(now())*" then the current year will be displayed, not the "**=year(now())**" text. If true, then the cell's entered value is displayed in the control. For example, if the cell contained "*=year(now())*" then the text "**=year(now())**" will be displayed not the current year.
click, change	No	None	The standard HTML attributes, **onClick** and **onChange,** are not available for use because they interfere with dynamic sheet updates. The attributes **click** and **change**

			replace **onClick** and **onChange**. These attributes provide the same functionality but are used differently. When in **interactive** mode, the attribute value for **click** or **change** *must* be the name of a JavaScript function that will return a **true** or **false** value. If the return value of the function is **true,** the spreadsheet will be updated with the HTML control value. If the return value of the function is false, the spreadsheet will not be updated with the HTML control value. If the mode is **batch**, the function will be passed-through to the standard HTML attributes **onClick** and **onChange**.
classname	No	None	This is to be used instead of the class passthrough attribute.

Pass-Through Attributes

The following optional attributes are passed through the JSP page generation phase without undergoing any changes. The attributes are not processed by the JSP page generation phase, but added to the generated **<input type="text">** tag exactly as they are entered. The attributes are expected to work as specified by the HTML 4.0 **<input type="text">** tag specification:

accept, accesskey, align, class (use classname instead), dir, disabled (use disabled = "true"), id, lang, maxlength, onblur, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, onfocus, onselect, readonly (use readonly="true"), size, style, tabindex, title

Example

```
<html>
<head>
<title>Date</title>
</head>
<body>
<h1>Date</h1>

<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       name="JSheetExamples"
       openbookname="/JSPCustomTagExamples/example.jss"
       sheet="0">

<jsheet:form>

<jsheet:date
       cell="R1C1"/>

</jsheet:form>
</jsheet:connect>

</body>
</html>
```

executequery

Use the **<executequery>** custom tag to execute a query on the current database session. Execute query will only execute a query. It will not place the result set in the JSheet workbook.

Syntax

<jsheet:executequery>
select * from myTable
</jsheet:executequery>

This tag pair must be nested inside a **<database> </database>** custom tag pair. The **<database> </database>** custom tag pair should be nested inside a **<connect> </connect>** custom tag pair.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
queryname	no	None	A valid query name. This must have been previously established in the **query** tag.

Pass-Through Attributes

This tag has no pass-through attributes.

Example

```
<html>
<head>
<title>ExecuteQuery</title>
</head>
<body>
<h1>ExecuteQuery</h1>

<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       name="JSheetExamples"
       openbookname="/JSPCustomTagExamples/example.jss"
       sheet="0">

<jsheet:database
       dbpassword="demo"
       dbname="myDatabase"
       dbuser="myDatabaseUser">

<jsheet:executequery>
Select * from myTable
</jsheet:executequery>

</jsheet:database>

</jsheet:connect>

</body>
</html>
```

executescript

Use the **<executescript>** custom tag to execute JavaScript on the JSServer. The body of the **<executescript>** tag is sent to the server to be executed.

Syntax

<jsheet:executescript>
  myJavaScript
</jsheet:executescript>

This tag pair must be nested inside a **<connect> </connect>** custom tag pair.

Attributes Processed by JSP

This tag has no attributes.

Pass-Through Attributes

This tag has no pass-through attributes.

Example

```
<html>
<head>
<title>ExecuteScript</title>
</head>
<body>
<h1>ExecuteScript</h1>

<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       name="JSheetExamples"
       openbookname="/JSPCustomTagExamples/example.jss"
       sheet="0">

<jsheet:executescript>
       myjavascript
</jsheet:executescript>

</jsheet:connect>

</body>
</html>
```

fetchinto

Use the **<fetchinto>** custom tag to place the results of a database query into a cell of a sheet.

Syntax

<jsheet:fetchinto
  [**queryname=**"*string*"]
  [**sheet=**"*string*"]
  cell="*string*"
  [{**mode=**"all" | "first" | "last"} | {**mode=**"count" **rows=**"*non-negative_int*"}]**>**
</jsheet:fetchinto>

This tag pair must be nested inside a **<database> </database>** custom tag pair. The **<datatase>** custom tag pair must be nested inside a **<connect> </connect>** custom tag pair.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
cell	Yes	None	The location in the spreadsheet to place the result set. This must be a fully qualified name formatted as: BookName.jss:SheetName!A2. A range may also be used instead of a single cell. If a range is used and the range is smaller than the result set, the result set will be truncated.
mode	No	all	The type of fetch to perform
queryname	No	None	The name of the query. This must be a valid query as defined by the query tag.
rows	No	all	The number of rows to return if **mode** has been specified as *"count"*
sheet	No	The sheet setting from the **<connect>** tag.	The sheet in the workbook where the result of the query will be placed.

Pass-Through Attributes

This tag has no pass-through attributes.

Example

```
<html>
<head>
<title>Fetchinto</title>
</head>
<body>
<h1>Fetchinto</h1>
<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       sheet="0"
       name="JSheetExamples"
       shared="true"
       openbookname="/JSPCustomTagExamples/examples.jss"
       timeout="2">

<jsheet:database
```

```
       dbuser="demo"
       dbpassword="demo"
       dbname="myDatabase">

<jsheet:fetchinto
       cell="A1">
       select * from myTable
</jsheet:fetchinto>

</jsheet:database>

</jsheet:connect>

</body>
</html>
```

form

Use the **<form> </form>** custom tag pair to generate an HTML form.

Syntax

<jsheet:form
  [**name=**"*string*"]
  [**action=**"*string*"]
  [**charset=**"*string*"]
  [**click=**"*string*"]
  [**classname=**"string"]
  [*pass through attributes*]**>**
</jsheet:form>

This tag pair must be nested inside a **<connect> </connect>** custom tag pair. The **<form> </form>** custom tag pair should surround all JSP custom tag input controls.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
name	No	Auto-generated as follows: if **<form>** is the second JSP custom tag on the current page, the generated HTML <form> attribute name is **tag2**.	The name attribute of the generated <form> tag.
action	No	Current JSP page	Indicates the URL to process after processing has completed for this page. Since there is no form submittal in **interactive** mode, this attribute is only available in **batch** and **palm** modes.
charset	No	UTF-8	This is to be used instead of accept-charset.
click	No	None	The standard HTML attribute, **onClick**, is not available for use since it interferes with dynamic sheet updates. The attribute **click** replaces **onClick**. This attribute provides the same functionality but it is used differently. When in **interactive** mode, the attribute value for **click** *must* be the name of a JavaScript function that returns a **true** or **false** value. If the return value of the function is **true,** the spreadsheet will be updated with the HTML control value. If the return value of the function is false, the spreadsheet will not be updated with the HTML control value. If the mode is **batch**, the attribute value defined will be placed into the

			standard HTML attribute **onClick**.
classname	No	None	This is to be used instead of the class passthrough attribute.

Pass-Through Attributes

The following optional attributes are passed through the JSP page generation phase without undergoing any changes. The attributes are not processed by the JSP page generation phase, but added to the generated **<form>** tag exactly as they are entered. The attributes are expected to work as specified by the HTML 4.0 **<form>** tag specification.

accept-charset (use charset instead), class (use classname instead), dir, enctype, id, lang, method, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, onreset, onsubmit, style, target, title

Example

```
<html>
<head>
<title>Form</title>
</head>
<body>
<h1>Form</h1>

<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       sheet="0"
       name="JSPExamples"
       shared="true"
       openbookname="/JSPCustomTagExamples/examples.jss"
       timeout="2">

<jsheet:form >
       action="/JSPCustomTagExamples/myNextJspPage.jsp"
</jsheet:form>

</jsheet:connect>
</body>
</html>
```

hidden

Use this tag to create a hidden input control on a form. A hidden tag is generally used to send information between the browser and server that you do not want displayed, such as HTML.

Syntax

<jsheet:hidden
  [**name=**"*string*"]
  [**sheet=**{"*string*" | "*nonnegative int*"}]
  cell="*string*"
  [**autorefresh=**"*boolean*"**/>**

This tag must be nested inside a **<form> </form>** custom tag pair, and the **<form> </form>** custom tag pair must be nested inside a **<connect> </connect>** custom tag pair.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
name	No	Auto-generated as follows: if **<hidden>** is the second JSP custom tag on the current page, the generated HTML **<input>** element is named **tag2**.	The name of the generated **<input>** control.
sheet	No	The value from the **sheet** attribute of the **<connect>** tag.	The sheet that contains the **cell** attribute's value. This value can be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
cell	Yes	none	The cell that contains the HTML to be sent. This value should be a cell reference (A1 or R1C1) or a range name that indicates one cell.
autorefresh	No	true	This attribute is only valid in **inter-active** mode. If this value is **true**, then the control is updated whenever the control's associated cell is changed in the applet. If this value is **false**, then the control is not updated whenever the control's associated cell is changed in the applet.

Pass-Through Attributes

This tag has no pass-through attributes.

Example

```
<html>
<head>
<title>Hidden</title>
</head>
<body>
```

```
<h1>Hidden</h1>

<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       sheet="0"
       name="JSheetExamples"
       shared="true"
       openbookname="/JSPCustomTagExamples/examples.jss"
       timeout="2">

<jsheet:form>

<jsheet:hidden
       sheet="0"
       cell="A7"/>

</jsheet:form>
</jsheet:connect>
</body>
</html>
```

ifmode

The <ifmode> </ifmode> custom tag pair should be used as a marker to indicate whether its body should be processed or skipped. The <ifmode> body is processed only if the specified mode is valid and is the current mode as specified in the <connect> tag.

Syntax

<jsheet:ifmode
  [**name=**"*string*"]
  [**mode=**"*string*"]
  cell="*string*"**/>**

This tag must be nested inside a **<connect> </connect>** custom tag pair.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
name	no	The default value is auto-generated as follows. If **<date>** is the second JSP custom tag on the current page, then the generated HTML **<input>** element would be named tag2.	Must be a valid JavaScript identifier - each character may be a letter, underscore, dollar sign, or digit, but the first character may not be a digit.
mode	Yes	None	If the mode from the connect tag matches the value of this attribute, the contents of the **<ifmode>** tag will be processed. Can contain one or more modes. The list is a vertical bar (|) delimited list. For example: "**batch|palm**". Valid values are **batch**, **palm** and **interactive**.

Pass-Through Attributes

This tag has no pass-through attributes.

Example

```
<html>
<head>
<title>Hidden</title>
</head>
<body>
<h1>Hidden</h1>

<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       sheet="0"
       name="JSheetExamples"
       shared="true"
       openbookname="/JSPCustomTagExamples/examples.jss"
       timeout="2">

<jsheet:ifmode>
```

```
       mode="interactive">
mode = interactive !!

<jsheet:ifmode
       mode="batch">
mode = batch !!

</jsheet:ifmode>
</jsheet:ifmode>

</jsheet:connect>
</body>
</html>
```

ifsubmitted / ifnotsubmitted

Use the **<ifsubmitted> </ifsubmitted>** tag pair when processing post methods. Use the **<ifnotsubmitted> </ifnotsubmitted>** custom tag pair to process get methods. Typically, these tag pairs are used to determine the processing of their event bodies when a form is submitted or not submitted. Since there is no form submittal in **interactive** mode, this custom tag pair is only available in **batch** and **palm** modes.

Syntax

<jsheet:ifsubmitted>
event body
</jsheet:ifsubmitted>

<jsheet:ifnotsubmitted>
event body
</jsheet:ifnotsubmitted>

This tag has no nesting requirements.

Attributes Processed by JSP

This tag has no attributes.

Pass-Through Attributes

This tag has no pass-through attributes.

Example

```
<html>
<head>
<title>If Submitted /If Not Submitted</title>
</head>
<body>
<h1>If Submitted/If Not Submitted</h1>

<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       sheet="0"
       name="JSheetExamples"
       shared="true"
       openbookname="/JSPCustomTagExamples/examples.jss"
       timeout="2">

<jsheet:form >

<jsheet:ifnotsubmitted>

<jsheet:hidden
       sheet="0"
       cell="A7"/>

</jsheet:ifnotsubmitted>

<jsheet:ifsubmitted>
<jsheet:button
       action="submit"/>

</jsheet:ifsubmitted>

</jsheet:form>
</jsheet:connect>
```

```
</body>
</html>
```

image

Use this tag to display an image.

Syntax

<jsheet:image
[**name=**"*string*"]
[**sheet=**{"*string*" | "*nonnegative int*"}]
cell="*string*"
[**autorefresh=**"*boolean*"]
[**click=**"*string*"]
[**classname=**"*string*"]
[*pass through attributes*]**/>**

This tag must be nested inside a **<connect> </connect>** custom tag pair.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
name	No	Auto-generated as follows: if **<img>** is the second JSP custom tag on the current page, the generated HTML **<img>** element is named **tag2**.	The name of the generated **<input>** control.
sheet	No	The value from the **sheet** attribute of the **<connect>** tag.	The sheet that contains the **cell** attribute's value. This value can be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
cell	Yes	none	The cell that contains the URL of the image to be displayed. This value should be a cell reference (A1 or R1C1) or a range name that indicate one cell.
autorefresh	No	true	This attribute is only valid in **interactive** mode. If this value is **true**, then the control is updated whenever the control's associated cell is changed in the applet. If this value is **false**, then the control is not updated whenever the control's associated cell is changed in the applet.
click	No	None	The standard HTML attribute, **onClick**, is not available for use since it interferes with dynamic sheet updates. The attribute **click** replaces **onClick**. This attribute provides the same functionality but is used differently. When in **interactive** mode, the attribute value for **click** *must* be the name of a JavaScript function that returns a **true** or **false** value. If the

			return value of the function is **true,** the spreadsheet will be updated with the HTML control value. If the return value of the function is false, the spreadsheet will not be updated with the HTML control value. If the mode is **batch**, the attribute value defined will be placed into the standard HTML attribute **onClick**.
classname	No	None	This is to be used instead of the class passthrough attribute.

Pass-Through Attributes

The following optional attributes are passed through the JSP page generation phase without undergoing any changes. The attributes are not processed by the JSP page generation phase, but added to the generated **<img>** tag exactly as they are entered. The attributes are expected to work as specified by the HTML 4.0 **<img>** tag specification:

align, alt, border, class (use classname instead), dir, height, hspace, id, ismap, lang, longdesc, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, style, title, usemap, vspace, width

Example

```
<html>
<head>
<title>Image</title>
</head>
<body>
<h1>Image</h1>

<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       sheet="0"
       name="JSheetExamples"
       shared="true"
       openbookname="/JSPCustomTagExamples/examples.jss"
       timeout="2">

<jsheet:form>

<jsheet:image
       sheet="0"
       cell="A8"/>

</jsheet:form>
</jsheet:connect>

</body>
</html>
```

listbox

Use this tag to create a list box control on a form.

Syntax

<jsheet:listbox
[**name=**"*string*"]
[**sheet=**{"*string*" | "*non-negative int*"}]
cell="*string*"
[**size=**"*int*>=*2*"]
[**autorefresh=**"*boolean*"]
[**updatesheet=**"*boolean*"]
{{**valuerange=**"*string*" [**valuesheet=**{"*string*" | "*non-negative int*"}]} | {**valuelist=**"*string*"}}
[{**labelrange=**"*string*"] [**labelsheet=**{"*string*" | "*non-negative int*"}]} | {**labellist=**"*string*"}]
[**click=**"*string*"]
[**change=**"*string*"]
[**classname=**"*string*"]
[*pass through attributes*]**/>**

This tag must be nested inside a **<form> </form>** custom tag pair, and the **<form> </form>** custom tag pair must be nested inside a **<connect> </connect>** custom tag pair.

valuerange and **valuelist** are mutually exclusive attributes. **labelrange** and **labellist** are also mutually exclusive attributes. The number of values in the **valuerange/valuelist** attribute takes precedence over the number of values in the **labelrange/labellist** attribute. Thus, the number of displayed labels is determined by the number of **valuerange/valuelist** values.

If the **valuerange/valuelist** attribute contains more values than the **labelrange/labellist** attribute, then the extra values in the **valuerange/valuelist** attribute are used as "fill-ins" for the missing labels from the list of labels in the **labelrange/labellist** attribute. For example, if there are six values in the **valuerange/valuelist** attribute and only four values in the **labelrange/labellist** attribute, then the last two values in the **valuerange/valuelist** attribute are used as the last two labels.

If the **valuerange/valuelist** attribute contains less values than the **labelrange/labellist** attribute, then the extra values in the **labelrange/labellist** attribute are ignored. For example, if there are ten values in the **valuerange/valuelist** attribute and eleven values in the **labelrange/labellist** attribute, then the last value in the **labelrange/labellist** attribute is not displayed.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
name	No	Auto-generated as follows: if **<listbox>** is the second JSP custom tag on the current page, the generated HTML **<select>** element is named **tag2**.	The name of the generated **<input>** control.
sheet	No	The value from the **sheet** attribute of the **<connect>** tag.	The sheet that contains the **cell** attribute's value. This value can be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
cell	Yes	none	The target cell to contain the selected listbox values. This value should be a cell reference (A1 or R1C1) or a range name that indicates one cell.

autorefresh	No	true	This attribute is only valid in **inter-active** mode. If this value is **true**, then the control is updated whenever the control's associated cell is changed in the applet. If this value is **false**, then the control is not updated whenever the control's associated cell is changed in the applet.
size	No	4	Specifies the height of the list box. This value should be an integer >= to 2.
valuerange	Only if the **valuesheet** attribute is specified.	none	The range that contains the listbox values. This value should be a cell range (e.g., A1, R1C1, or A1..D4) or a range name. The range size should match the range size specified for the associated **labelrange** attribute. If the **labelrange** attribute is not defined, the **valuerange** values are displayed as the listbox items. The sheet that contains this range is specified by the **valuesheet** attribute. If **valuesheet** is not defined, the **sheet** attribute value is used. If the **sheet** attribute is not defined, the value is taken from the connect tag's **sheet** attribute.
valuesheet	Only if the **valuerange** attribute is specified.	The value of the **sheet** attribute.	The sheet that contains the **valuerange**'s attribute value. This value can be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
valuelist	Only if the **valuerange** and **valuesheet** attributes are not specified.	none	The list of listbox values. Use a vertical bar (|) delimited list. For example: "house|car|business" The number of values specified here should match the number of values specified for the associated **labellist** attribute. If the **labellist** attribute is not defined, the **valuelist** values are displayed as the listbox items.
labelrange	Only if the **labelsheet** attribute is specified.	The value of the **valuerange** attribute.	The range that contains the labels to be displayed as the listbox items. This value should be a cell range (e.g., A1, R1C1, or A1..D4) or a range name. The range size should match the range size specified for the associated **valuerange** attribute. The sheet that contains this range is

			specified by the **labelsheet** attribute. If **labelsheet** is not defined, the **sheet** attribute value is used. If the **sheet** attribute is not defined, the value is taken from the connect tag's **sheet** attribute.
labelsheet	Only if the **labelrange** attribute is specified.	The value of the **valuesheet** attribute.	The sheet that contains the **label-range**'s attribute value. This value can be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
labellist	Only if the **labelrange** and **labelsheet** attributes are not specified.	none	A list of labels to be displayed as the listbox items. Use a vertical bar (|) delimited list. For example: "house|car|business" The number of values specified here should match the number of values specified for the associated **valuelist** attribute.
updatesheet	No	true	This attribute is only valid in **inter-active** mode. If this value is **true**, then the control's associated applet cell is updated whenever the control is changed. If this value is **false**, then the control's associated applet cell is not updated whenever the control is changed.
click, change	No	None	The standard HTML attributes, **onClick** and **onChange,** are not available for use because they interfere with dynamic sheet updates. The attributes **click** and **change** replace **onClick** and **onChange**. These attributes provide the same functionality but are used differently. When in **interactive** mode, the attribute value for **click** or **change** *must* be the name of a JavaScript function that will return a **true** or **false** value. If the return value of the function is **true,** the spreadsheet will be updated with the HTML control value. If the return value of the function is false, the spreadsheet will not be updated with the HTML control value. If the mode is **batch**, the function will be passed-through to the standard HTML attributes **onClick** and **onChange**.
classname	No	None	This is to be used instead of the class passthrough attribute.

Pass-Through Attributes

The following optional attributes are passed through the JSP page generation phase without undergoing any changes. The attributes are not processed by the JSP page generation phase, but are added to the generated **<select>** tag exactly as they are entered. The attributes are expected to work as specified by the HTML 4.0 **<select>** tag specification:

class (use classname instead), dir, disabled (use disabled="true"), id, lang, onblur, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, size, style, tabindex, title

Example

```
<html>
<head>
<title>ListBox</title>
</head>
<body>
<h1>ListBox</h1>

<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       sheet="0"
       name="JSheetExamples"
       shared="true"
       openbookname="/JSPCustomTagExamples/examples.jss"
       timeout="2">

<jsheet:form>

<jsheet:listbox
       sheet="0"
       cell="A10"
       valuelist="eastsales|westsales|southsales"/>

</jsheet:form>
</jsheet:connect>

</body>
</html>
```

password

Use this tag to create a single password input control on a form.

Syntax

<jsheet:password
[**name=***"string"*]
[**sheet=**{*"string"* | *"nonnegative int"*}]
cell=*"string"*
[**updatesheet=**"*boolean*"]
[**click=**"*string*"]
[**change=**"*string*"]
[**classname=**"*string*"]
[*pass through attributes*]**/>**

This tag must be nested inside a **<form> </form>** custom tag pair, and the **<form> </form>** custom tag pair must be nested inside a **<connect> </connect>** custom tag pair.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
name	No	Auto-generated as follows: if **<password>** is the second JSP custom tag on the current page, the generated HTML **<input>** element is named **tag2**.	The name of the generated **<input>** control.
sheet	No	The value from the **sheet** attribute of the **<connect>** tag.	The sheet that contains the **cell** attribute's value. This value can be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
cell	Yes	none	The target cell to contain the password. If the control contains a string, the cell displays asterisks. Otherwise, the cell is empty. This value should be a cell reference (A1 or R1C1) or a range name that indicates one cell.
updatesheet	No	true	This attribute is only valid in **interactive** mode. If this value is **true**, then the control's associated applet cell is updated whenever the control is changed. If this value is **false**, then the control's associated applet cell is not updated whenever the control is changed.
click, change	No	None	The standard HTML attributes, **onClick** and **onChange,** are not available for use because they interfere with dynamic sheet updates. The attributes **click** and **change** replace **onClick** and **onChange**. These attributes provide the same functionality but are used differently.

			When in **interactive** mode, the attribute value for **click** or **change** *must* be the name of a JavaScript function that will return a **true** or **false** value. If the return value of the function is **true,** the spreadsheet will be updated with the HTML control value. If the return value of the function is false, the spreadsheet will not be updated with the HTML control value. If the mode is **batch**, the function will be passed-through to the standard HTML attributes **onClick** and **onChange**.
classname	No	None	This is to be used instead of the class passthrough attribute.

Pass-Through Attributes

The following optional attributes are passed through the JSP page generation phase without undergoing any changes. The attributes are not processed by the JSP page generation phase, but added to the generated **<input type="password">** tag exactly as they are entered. The attributes are expected to work as specified by the HTML 4.0 **<input type="password">** tag specification:

accept, accesskey, align, class, dir, disabled (use disabled="true"), id, lang, maxlength, onblur, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, onselect, readonly (use readonly="true"), size, style, tabindex, title

Example

```
<html>
<head>
<title>Password</title>
</head>
<body>
<h1>Password</h1>

<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       sheet="0"
       name="JSheetExamples"
       shared="true"
       openbookname="/JSPCustomTagExamples/examples.jss"
       timeout="2">

<jsheet:form >

<jsheet:password
       cell="A10"/>

</jsheet:form>
</jsheet:connect>

</body>
</html>
```

query

Use the **<query>** custom tag to establish a relationship between a specific query and a queryname. The **<query> </query>** tag pair contains the SQL statement for the query.

Syntax

<jsheet:query
queryname="*string*"**>**
*select * from myTable*
</jsheet:query>

This tag pair must be nested inside a **<connect> </connect>** custom tag pair.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
queryname	Yes	None	The name of the query.

Pass-Through Attributes

This tag has no pass-through attributes.

Example

```
<html>
<head>
<title>Query</title>
</head>
<body>
<h1>Query</h1>

<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       name="JSheetExamples"
       openbookname="/JSPCustomTagExamples/example.jss"
       sheet="0">

<jsheet:database
       dbpassword="demo"
       dbname="myDatabase"
       dbuser="myDatabaseUser">

<jsheet:query
       queryname="SelectAll">
select * from myTable
</jsheet:query>

<jsheet:executequery
       queryname="SelectAll"/>

</jsheet:database>

</jsheet:connect>

</body>
</html>
```

radio

Use this tag to create a group of radio button input controls on a form.

Syntax

<jsheet:radio
[**name=**"*string*"]
[**sheet=**{"*string*" | "*non-negative int*"}]
cell="*string*"
[**autorefresh=**"*boolean*"]
[**updatesheet=**"*boolean*"]
{{**valuerange=**"*string*" [**valuesheet=**{"*string*" | "*non-negative int*"}]} | {**valuelist=**"*string*"}}
[{**labelrange=**"*string*"]
[**labelsheet=**{"*string*" | "*non-negative int*"}]} | {**labellist=**"*string*"}]
[**click=**"*string*"]
[**change=**"*string*"]
[**classname=**"*string*"]
[*pass through attributes*]**/>**

This tag must be nested inside a **<form> </form>** custom tag pair, and the **<form> </form>** custom tag pair must be nested inside a **<connect> </connect>** custom tag pair.

valuerange and **valuelist** are mutually exclusive attributes. **labelrange** and **labellist** are also mutually exclusive attributes. The number of values in the **valuerange**/**valuelist** attribute takes precedence over the number of values in the **labelrange**/**labellist** attribute. Thus, the number of displayed labels is determined by the number of **valuerange**/**valuelist** values.

If the **valuerange**/**valuelist** attribute contains more values than the **labelrange/labellist** attribute, then the extra values in the **valuerange/valuelist** attribute are used as "fill-ins" for the missing labels from the list of labels in the **labelrange/labellist** attribute. For example, if there are six values in the **valuerange/valuelist** attribute and only four values in the **labelrange/labellist** attribute, then the last two values in the **valuerange/valuelist** attribute are used as the last two labels.

If the **valuerange/valuelist** attribute contains less values than the **labelrange/labellist** attribute, then the extra values in the **labelrange/labellist** attribute are ignored. For example, if there are ten values in the **valuerange/valuelist** attribute and eleven values in the **labelrange/labellist** attribute, then the last value in the **labelrange/labellist** attribute is not displayed.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
name	No	Auto-generated as follows: if **<radio>** is the second JSP custom tag on the current page, the generated HTML **<input>** element is named **tag2**.	The name of the generated **<input>** control.
sheet	No	The value from the **sheet** attribute of the **<connect>** tag.	The sheet that contains the **cell** attribute's value. This value can be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
cell	Yes	none	The target cell to contain the selected radio button value. This value should be a cell reference (A1 or R1C1) or a range name that indicates one cell.

autorefresh	No	true	This attribute is only valid in **inter-active** mode. If this value is **true**, then the control is updated whenever the control's associated cell is changed in the applet. If this value is **false**, then the control is not updated whenever the control's associated cell is changed in the applet.
valuerange	Only if the **valuesheet** attribute is specified.	none	The range that contains the radio button values. This value should be a cell range (e.g., A1, R1C1, or A1..D4) or a range name. The range size should match the range size specified for the associated **labelrange** attribute. If the **labelrange** attribute is not defined, the **valuerange** values are displayed as the radio button text. The sheet that contains this range is specified by the **valuesheet** attribute. If **valuesheet** is not defined, the **sheet** attribute value is used. If the **sheet** attribute is not defined, the value is taken from the connect tag's **sheet** attribute.
valuesheet	Only if the **valuerange** attribute is specified.	The value of the **sheet** attribute.	The sheet that contains the **valuerange**'s attribute value. This value can be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
valuelist	Only if the **valuerange** and **valuesheet** attributes are not specified.	none	The list of radio button values. Use a vertical bar (|) delimited list. For example: "house|car|business" The number of values specified here should match the number of values specified for the associated **labellist** attribute. If the **labellist** attribute is not defined, the **valuelist** values are displayed as the listbox items.
labelrange	Only if the **labelsheet** attribute is specified.	The value of the **valuerange** attribute.	The range that contains the labels to be displayed as the radio button text. This value should be a cell range (e.g., A1, R1C1, or A1..D4) or a range name. The range size should match the range size specified for the associated **valuerange** attribute. The sheet that contains this range is specified by the **labelsheet** attribute. If **labelsheet** is not defined, the **sheet** attribute value is used. If the **sheet** attribute is not defined, the value is

			taken from the connect tag's **sheet** attribute.
labelsheet	Only if the **labelrange** attribute is specified.	The value of the **valuesheet** attribute.	The sheet that contains the **label-range**'s attribute value. This value can be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
labellist	Only if the **labelrange** and **labelsheet** attributes are not specified.	none	A list of labels to be displayed as the radio button text. Use a vertical bar (|) delimited list. For example: "house|car|business" The number of values specified here should match the number of values specified for the associated **valuelist** attribute.
updatesheet	No	true	This attribute is only valid in **inter-active** mode. If this value is **true**, then the control's associated applet cell is updated whenever the control is changed. If this value is **false**, then the control's associated applet cell is not updated whenever the control is changed.
click, change	No	None	The standard HTML attributes, **onClick** and **onChange,** are not available for use because they interfere with dynamic sheet updates. The attributes **click** and **change** replace **onClick** and **onChange**. These attributes provide the same functionality but are used differently. When in **interactive** mode, the attribute value for **click** or **change** *must* be the name of a JavaScript function that will return a **true** or **false** value. If the return value of the function is **true,** the spreadsheet will be updated with the HTML control value. If the return value of the function is false, the spreadsheet will not be updated with the HTML control value. If the mode is **batch**, the function will be passed-through to the standard HTML attributes **onClick** and **onChange**.
classname	No	None	This is to be used instead of the class pass-through attribute.

Pass-Through Attributes

The following optional attributes are passed through the JSP page generation phase without undergoing any changes. The attributes are not processed by the JSP page generation phase, but added to the generated **<input**

type="radio"> tag exactly as they are entered. The **<radio>** custom tag generates a group of HTML **<input>** tags. The specified pass-through attributes are added to all of the **<input>** tags in the group:

accept, accesskey, align, class (use classname instead), dir, disabled (use disabled="true"), id, lang, onblur, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, size, style, tabindex, title

Example

```
<html>
<head>
<title>RadioButton</title>
</head>
<body>
<h1>RadioButton</h1>

<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       sheet="0"
       name="JSheetExamples"
       shared="true"
       openbookname="/JSPCustomTagExamples/examples.jss"
       timeout="2">

<jsheet:form >

<jsheet:radio
       sheet="0"
       cell="A13"
       labelrange="M12..M15"
       labelsheet="0"
       valuerange="P12..P15"
       valuesheet="0"/>

</jsheet:form>
</jsheet:connect>

</body>
</html>
```

static

Use this tag to create a read-only field that displays text.

Syntax

<jsheet:static
[**name=**"*string*"]
[**sheet=**{"*string*" | "*non-negative int*"}]
cell="*string*"
[**click=**"*string*"]
[**classname**="*string*"]
[*pass-through attributes*]**/>**

This tag must be nested inside a **<connect> </connect>** custom tag pair.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
name	No	Auto-generated as follows: if **<static>** is the second JSP custom tag on the current page, the generated HTML element is named **tag2**.	The name of the generated **<input>** control.
sheet	No	The value from the **sheet** attribute of the **<connect>** tag.	The sheet that contains the **cell** attribute's value. This value can be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
cell	Yes	none	The cell that contains the text to be displayed. This value should be a cell reference (A1) or a range name.
click	No	None	The standard HTML attribute, **onClick**, is not available for use since it interferes with dynamic sheet updates. The attribute **click** replaces **onClick**. This attribute provides the same functionality but it is used differently. When in **interactive** mode, the attribute value for **click** *must* be the name of a JavaScript function that returns a **true** or **false** value. If the return value of the function is **true,** the spreadsheet will be updated with the HTML control value. If the return value of the function is false, the spreadsheet will not be updated with the HTML control value. If the mode is **batch**, the attribute value defined will be placed into the standard HTML attribute **onClick**.
classname	No	None	This is to be used instead of the class passthrough attribute.

Pass-through Attributes

dir, class (use classname instead), id, lang, style, title, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup

Example

```
<html>
<head>
<title>Static</title>
</head>
<body>
<h1>Static</h1>

<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       sheet="0"
       name="JSheetExamples"
       shared="true"
       openbookname="/JSPCustomTagExamples/examples.jss"
       timeout="2">

<jsheet:form>

<jsheet:static
       sheet="0"
       cell="A25"/>

</jsheet:form>
</jsheet:connect>

</body>
</html>
```

table

Use this tag to create a table with data from a sheet.

Syntax

<jsheet:table
  [**name=**"*string*"]
  range="*string*"
  [**autorefresh=**"*boolean*"]
  [**updatesheet=**"*boolean*"]
  [**sheet=**{"*string*" | "*nonnegative int*"}]
  [**editablecells=**"*string*"]
  [**formattedcells=**"*boolean*"]
  [**click=**"*string*"]
  [**classname=**"*string*"]
  [*pass through attributes*]**/>**

The generated table is in the form of an HTML **<table>** element. HTML formatting tags are generated for each table cell such that they correspond to the format in the grid's associated cells. The table is generated automatically without explicitly having to define rows or columns.

This tag must be nested inside a **<connect> </connect>** custom tag pair. In addition, if the **editablecells** attribute contains a non-empty string value, this tag must be nested inside a **<form> </form>** custom tag pair.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
name	No	Auto-generated as follows: if **<table>** is the second JSP custom tag on the current page, the generated HTML element is named **tag2**.	The name of the generated **<table>** element.
range	No	The smallest top-left range that contains all the non-blank cells.	The range of data to display in the table. This value should be a cell range (e.g., A1, R1C1, or A1..D4) or a range name.
sheet	No	The value from the **sheet** attribute of the **<connect>** tag.	The sheet that contains the **range** attribute's value. This value can be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
autorefresh	No	true	This attribute is only valid in **inter-active** mode. If this value is **true**, then the control is updated whenever the control's associated cell is changed in the applet. If this value is **false**, then the control is not updated whenever the control's associated cell is changed in the applet.
editablecells	No	none	The cells to be editable in the generated table. The edits are written to the sheet and the sheet is updated.

			This value must be a range of cells, a comma-delimited list of ranges, or a named range. Example: "r1c1" or "a1...d3, r1...t4"
formattedcells	No	true	HTML formatting tags are generated for each cell in the resulting table. The formats applied to the table are taken from the corresponding formats in the grid of the sheet. Formats from the sheet that are not supported by HTML are not generated, such as dashed line borders.
updatesheet	No	true	This attribute is only valid in **interactive** mode. If this value is **true**, then the control's associated applet cell is updated whenever the control is changed. If this value is **false**, then the control's associated applet cell is not updated whenever the control is changed.
click	No	None	The standard HTML attribute, **onClick**, is not available for use since it interferes with dynamic sheet updates. The attribute **click** replaces **onClick**. This attribute provides the same functionality but it is used differently. When in **interactive** mode, the attribute value for **click** *must* be the name of a JavaScript function that returns a **true** or **false** value. If the return value of the function is **true,** the spreadsheet will be updated with the HTML control value. If the return value of the function is false, the spreadsheet will not be updated with the HTML control value. If the mode is **batch**, the attribute value defined will be placed into the standard HTML attribute **onClick**.
classname	No	None	This is to be used instead of the class passthrough attribute.

Pass-Through Attributes

The following optional attributes are passed through the JSP page generation phase without undergoing any changes. The attributes are not processed by the JSP page generation phase, but added to the generated tag exactly as they are entered. The **<table>** tag generates an HTML **<table>** tag. Use the pass-through attributes as specified by the HTML 4.0 **<table>** tag specification:

align, bgcolor, border, cellpadding, cellspacing, class (use classname instead), dir, frame, id, lang, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, rules, style, summary, title, width

Example

```
<html>
<head>
<title>Table</title>
</head>
<body>
<h1>Table</h1>

<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       sheet="0"
       name="JSheetExamples"
       shared="true"
       openbookname="/JSPCustomTagExamples/examples.jss"
       timeout="2">

<jsheet:table
       range="K5..K15"
       sheet="0"
       formattedcells="true"/>

</jsheet:connect>

</body>
</html>
```

text

Use this tag to create a text input control on a form.

Syntax

<jsheet:text
[**name=**"*string*"]
[**sheet=**{"*string*" | "*nonnegative int*"}]
cell="*string*"
[**autorefresh**="*boolean*"]
[**updatesheet**="*boolean*"]
[**showcellentry=**"*boolean*"]
[**click=**"*string*"]
[**change=**"*string*"]
[**classname**="*string*"]
[*pass through attributes*]**/>**

This tag must be nested inside a **<form> </form>** custom tag pair, and the **<form> </form>** custom tag pair must be nested inside a **<connect> </connect>** custom tag pair.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
name	No	Auto-generated as follows: if **<text>** is the second JSP custom tag on the current page, the generated HTML **<input>** element is named **tag2**.	The name of the generated **<input>** control.
sheet	No	The value from the **sheet** attribute of the **<connect>** tag.	The sheet that contains the **cell** attribute's value.This value can be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
cell	Yes	none	The cell that contains the text to be displayed in the control. This value should be a cell reference (A1 or R1C1) or a range name that indicate one cell.
autorefresh	No	true	This attribute is only valid in **inter-active** mode. If this value is **true**, then the control is updated whenever the control's associated cell is changed in the applet. If this value is **false**, then the control is not updated whenever the control's associated cell is changed in the applet.
showcellentry	No	false	If this value is **false**, the cell's display value is displayed in the control. For example, if the cell contained "=year(now())," the current year is displayed, not the "=year(now())" text. If this value is **true**, the cell's entered value is displayed in the control. For

			example, if the cell contained "=year(now())," the "=year(now())" text is displayed, not the current year.
updatesheet	No	true	This attribute is only valid in **interactive** mode. If this value is **true**, then the control's associated applet cell is updated whenever the control is changed. If this value is **false**, then the control's associated applet cell is not updated whenever the control is changed.
click, change	No	None	The standard HTML attributes, **onClick** and **onChange,** are not available for use because they interfere with dynamic sheet updates. The attributes **click** and **change** replace **onClick** and **onChange**. These attributes provide the same functionality but are used differently. When in **interactive** mode, the attribute value for **click** or **change** *must* be the name of a JavaScript function that will return a **true** or **false** value. If the return value of the function is **true,** the spreadsheet will be updated with the HTML control value. If the return value of the function is false, the spreadsheet will not be updated with the HTML control value. If the mode is **batch**, the function will be passed-through to the standard HTML attributes **onClick** and **onChange**.
classname	No	None	This is to be used instead of the class passthrough attribute.

Pass-Through Attributes

The following optional attributes are passed through the JSP page generation phase without undergoing any changes. The attributes are not processed by the JSP page generation phase, but added to the generated **<input type="text">** tag exactly as they are entered. The attributes are expected to work as specified by the HTML 4.0 **<input type="text">** tag specification:

accesskey, align, class (use classname instead), dir, disabled (use disabled="true"), id, lang, maxlength, onblur, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, onselect, readonly (use readonly="true"), size, style, tabindex, title

Example

```
<html>
<head>
<title>Text</title>
</head>
<body>
<h1>Text</h1>

<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>
```

```
<jsheet:connect
       sheet="0"
       name="JSheetExamples"
       shared="true"
       openbookname="/JSPCustomTagExamples/examples.jss"
       timeout="2">

<jsheet:form>

<jsheet:text
       sheet="0"
       cell="A30"
       showcellentry="false"/>

</jsheet:form>
</jsheet:connect>

</body>
</html>
```

textarea

Use this tag to create a text area input control on a form.

Syntax

<jsheet:textarea
[**name=**"*string*"]
[**sheet=**{"*string*" | "*nonnegative int*"}]
cell="*string*"
[**autorefresh**="*boolean*"]
[**updatesheet**="*boolean*"]
[**showcellentry=**"*boolean*"]
[**click=**"*string*"]
[**change=**"*string*"]
[**classname**=”*string*”]
[pass through attributes]**/>**

This tag must be nested inside a **<form> </form>** custom tag pair, and the **<form> </form>** custom tag pair must be nested inside a **<connect> </connect>** custom tag pair.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
name	No	Auto-generated as follows: if **<textarea>** is the second JSP custom tag on the current page, the generated HTML **<textarea>** element is named **tag2**.	The name of the generated **<input>** control.
sheet	No	The value from the **sheet** attribute of the **<connect>** tag.	The sheet that contains the **cell**’s attribute value. This value can be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
cell	Yes	none	The cell that contains the text to be displayed in the control. This value should be a cell reference (A1) or a range name where the range contains only one cell.
autorefresh	No	true	This attribute is only valid in **inter-active** mode. If this value is **true**, then the control is updated whenever the control’s associated cell is changed in the applet. If this value is **false**, then the control is not updated whenever the control’s associated cell is changed in the applet.
showcellentry	No	false	If this value is **false**, the cell's display value is displayed in the control. For example, if the cell contained "=year(now())," the current year is displayed, not the "=year(now())" text.

			If this value is **true**, the cell's entered value is displayed in the control. For example, if the cell contained "=year(now())," the "=year(now())" text is displayed, not the current year.
updatesheet	No	true	This attribute is only valid in **interactive** mode. If this value is **true**, then the control's associated applet cell is updated whenever the control is changed. If this value is **false**, then the control's associated applet cell is not updated whenever the control is changed.
click, change	No	None	The standard HTML attributes, **onClick** and **onChange,** are not available for use because they interfere with dynamic sheet updates. The attributes **click** and **change** replace **onClick** and **onChange**. These attributes provide the same functionality but are used differently. When in **interactive** mode, the attribute value for **click** or **change** *must* be the name of a JavaScript function that will return a **true** or **false** value. If the return value of the function is **true,** the spreadsheet will be updated with the HTML control value. If the return value of the function is false, the spreadsheet will not be updated with the HTML control value. If the mode is **batch**, the function will be passed-through to the standard HTML attributes **onClick** and **onChange**.
classname	No	None	This is to be used instead of the class passthrough attribute.

Pass-Through Attributes

The following optional attributes are passed through the JSP page generation phase without undergoing any changes. The attributes are not processed by the JSP page generation phase, but added to the generated **<textarea>** tag exactly as they are entered. The attributes are expected to work as specified by the HTML 4.0 **<textarea>** tag specification:

accesskey, class (use classname instead), cols, dir, disabled (use disabled="true"), id, lang, onblur, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, onselect, readonly (use readonly="true"), rows, style, tabindex, title

Example

```
<html>
<head>
<title>TextArea</title>
</head>
<body>
```

```
<h1>TextArea</h1>

<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       sheet="0"
       name="JSheetExamples"
       shared="true"
       openbookname="/JSPCustomTagExamples/examples.jss"
       timeout="2">

<jsheet:form>

<jsheet:textarea
       sheet="0"
       cell="A35"
       showcellentry="false"/>

</jsheet:form>
</jsheet:connect>

</body>
</html>
```

time

Use this tag to create a date input control on a form in either a web page or a Palm device.

Syntax

<jsheet:time
  [**name=**"*string*"]
  [**sheet=**{"*string*" | "*nonnegative int*"}]
  cell="*string*"
  [**autorefresh**="*boolean*"]
  [**updatesheet**="*boolean*"]
  [**showcellentry=**"*boolean*"]
  [**click=**"*string*"]
  [**change=**"*string*"]
  [**classname**="*string*"]
  [pass through attributes]**/>**

This tag must be nested inside a **<form> </form>** custom tag pair, and the **<form> </form>** custom tag pair must be nested inside a **<connect> </connect>** custom tag pair.

Attributes Processed by JSP

Attribute	Required	Default Value	Description
name	No	Auto-generated as follows: if **<time>** is the second JSP custom tag on the current page, the generated HTML **<time>** element is named **tag2**.	The name of the generated **<input>** control.
sheet	No	The value from the **sheet** attribute of the **<connect>** tag.	The sheet that contains the **cell**'s attribute value. This value can be the index or the name of the sheet. Sheet indexes start at 0 for the first sheet.
cell	Yes	none	The cell that contains the text to be displayed in the control. This value should be a cell reference (A1) or a range name where the range contains only one cell.
autorefresh	No	true	This attribute is only valid in **interactive** mode. If this value is **true**, then the control is updated whenever the control's associated cell is changed in the applet. If this value is **false**, then the control is not updated whenever the control's associated cell is changed in the applet.
showcellentry	No	false	If this value is **false**, the cell's display value is displayed in the control. For example, if the cell contained "=year(now())," the current year is displayed, not the "=year(now())" text. If this value is **true**, the cell's entered

			value is displayed in the control. For example, if the cell contained "=year(now())," the "=year(now())" text is displayed, not the current year.
updatesheet	No	true	This attribute is only valid in **interactive** mode. If this value is **true**, then the control's associated applet cell is updated whenever the control is changed. If this value is **false**, then the control's associated applet cell is not updated whenever the control is changed.
click, change	No	None	The standard HTML attributes, **onClick** and **onChange,** are not available for use because they interfere with dynamic sheet updates. The attributes **click** and **change** replace **onClick** and **onChange**. These attributes provide the same functionality but are used differently. When in **interactive** mode, the attribute value for **click** or **change** *must* be the name of a JavaScript function that will return a **true** or **false** value. If the return value of the function is **true,** the spreadsheet will be updated with the HTML control value. If the return value of the function is false, the spreadsheet will not be updated with the HTML control value. If the mode is **batch**, the function will be passed-through to the standard HTML attributes **onClick** and **onChange**.
classname	No	None	This is to be used instead of the class passthrough attribute.

Pass-Through Attributes

The following optional attributes are passed through the JSP page generation phase without undergoing any changes. The attributes are not processed by the JSP page generation phase, but added to the generated **<input>** tag exactly as they are entered. The attributes are expected to work as specified by the HTML 4.0 **<input>** tag specification:

accept, accesskey, align, class (use classname instead), dir, disabled (use disabled="true"), id, lang, maxlength, onblur, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, onselect, readonly (use readonly="true"), size, style, tabindex, title.

Example

```
<html>
<head>
<title>Time</title>
</head>
<body>
<h1>Time</h1>
```

```
<%@ taglib prefix="jsheet" uri="/JSheetExamples" %>

<jsheet:connect
       sheet="0"
       name="JSheetExamples"
       shared="true"
       openbookname="/JSPCustomTagExamples/examples.jss"
       timeout="2">

<jsheet:form>

<jsheet:time
       sheet="0"
       cell="A35"/>

</jsheet:form>
</jsheet:connect>

</body>
</html>
```