



LiteWebServer 3

JSTL Module Reference Manual

2003-02-24, Version 1.0

INTRODUCTION	2
LICENSES	2
CONTACT INFORMATION	2
FEATURES	2
INSTALLATION	3
JDBC DRIVERS FOR JSTL	3
JSTL LIBRARY URIS AND DEFAULT PREFIXES	3
CONFIGURATION OPTIONS	4
Database Access Library Settings	4
I18N and Formatting Library Settings	5

Thank you for choosing LiteWebServer™ (LWS), a small and easy-to-use Java web server with native support for the Servlet API. It's small size (roughly 1 MB or less, depending on Java version and optional features) makes it ideal for embedding in other Java applications, for bundling with web application demos sent to potential customers, and for running web applications distributed on CDs. It's also ideal for Java web application development, personal use or a small intranet.

This Reference Manual describes how to configure and use the JSTL Module. Please see the Base Module Reference Manual for how to install, configure and use LiteWebServer in general and the JSP Module Reference Manual for general JSP configuration issues. For information about how to develop JSP pages using JSP Standard Tag Library (JSTL), please use the resources listed on Sun's JSTL pages, <http://java.sun.com/products/jsp/jstl/>. We also recommend Hans Bergsten's *JavaServer Pages* (O'Reilly) if you want to learn more about JSP and JSTL.

Introduction

LiteWebServer (LWS) is a pure Java web server with native support for the Java Servlet API. It's a modular server, with a base module that contains the main functionality and add-on modules for additional features. The JSTL Module makes an implementation of the JSP Standard Tag Library (JSTL) available to all web applications served by LWS.

Licenses

LWS is licensed under a BSD-style open source license. Briefly, this means that you can use the product any way you want as long as you keep all copyright notices and include a note about where it comes from if you redistribute LWS with other software. See the *license.txt* file in the installation directory for details.

The JSTL Module includes software developed by the Apache Software Foundation (<http://www.apache.org/>). More specifically, it contains the JSTL 1.0 implementation developed as the Standard tag library in the Apache Taglibs project. See the Apache Software License, Version 1.1 (*Tomcat-license.txt*) in the installation directory for licensing terms with respect to the bundled ASF code.

Contact information

You can reach Gefion Software through one of the following mail addresses.

info@gefionsoftware.com for general questions or comments about the company and the products.

support@gefionsoftware.com to get help or report a problem with a product. Ideas about new features or other improvements are of course also welcome.

sales@gefionsoftware.com for questions about licenses.

Visit our web site at <http://www.gefionsoftware.com/> for up-to-date information.

Features

The LiteWebServer JSTL Module is a complete implementation of the JSP Standard Tag Library (JSTL) 1.0 specification, in other words, the following main features are available:

- Core tag library, with custom actions for conditional processing, iterations, importing data from external and internal resources, and more.
- XML tag library, with custom actions for XML parsing, transformation and accessing elements of a parsed XML document.

- Internationalization (I18N) and formatting library, with custom actions for formatting and parsing localized numbers and dates, using localized text, and more.
- Relational database access (SQL) library, with custom actions for reading from and writing to a relational database.
- Expression Language for easy access to application and request data.

Installation

The JSTL Module must be installed using the JustGetIt module manager application bundled with the LiteWebServer (LWS) Base Module, see the Base Module Reference Manual for details. The JustGetIt module manager also installs the correct version of the JSP Module if it's not already installed when you install the JSTL Module.

Installing the JSTL Module using the JustGetIt module manager places files in these directories under the LWS home directory (referred to as `$LWS_HOME` in some parts of this document):

etc

Files with information about the installed modules.

shared/lib

The JAR files that make up the JSTL implementation.

JDBC Drivers for JSTL

The JSTL SQL library uses JDBC to access your database. Most database vendors make a JDBC driver available to their customers for no extra charge. Sun also maintains a list of third-party drivers at <http://industry.java.sun.com/products/jdbc/drivers>.

Note! In order for the JSTL SQL tag library to find the JDBC driver, you *must* install the JAR files for the driver in the *shared/lib* directory.

JSTL Library URIs and Default Prefixes

The JSTL libraries come in two flavors: one set that accepts Expression Language (EL) expressions as action element attributes (the EL libraries) and one set that accepts request-time attribute values (the RT libraries). Except for the type of expressions they accept, they are identical in every way.

The following table lists the URIs and default prefixes for the EL libraries:

Library	URI	Prefix
Core	http://java.sun.com/jstl/core	c
XML Processing	http://java.sun.com/jstl/xml	x
I18N Formatting	http://java.sun.com/jstl/fmt	fmt
Database Access	http://java.sun.com/jstl/sql	sql

The following table lists the URIs and default prefixes for the RT libraries:

Library	URI	Prefix
Core	http://java.sun.com/jstl/core_rt	c_rt
XML Processing	http://java.sun.com/jstl/xml_rt	x_rt
I18N Formatting	http://java.sun.com/jstl/fmt_rt	fmt_rt

Database Access	http://java.sun.com/jstl/sql_rt	sql_rt
-----------------	---	--------

Configuration Options

Some default values can be defined as context initialization parameters in the deployment descriptor (*web.xml* file) for an application, e.g.:

```
<web-app>
  <context-param>
    <param-name>
      javax.servlet.jsp.jstl.sql.dataSource
    </param-name>
    <param-value>
      jdbc/Example
    </param-value>
  </context-param>
  ...
</web-app>
```

Defaults can also be set by a servlet or listener through the JSTL Config class using the constants described below. For details about these settings, see the JSTL specification available on Sun's JSTL pages, <http://java.sun.com/products/jsp/jstl/>, or Hans Bergsten's *JavaServer Pages* (O'Reilly).

Database Access Library Settings

A default for the `dataSource` attribute can be defined through the data source setting. It can be set as a `String` in this format, where optional parts are embedded in brackets:

```
url [, [driver] [, [user] [, [password]]]
```

This type of value is used to create a simple `DataSource` without any pooling capabilities and is only intended for prototype and low-end applications. It can also be set to a JNDI path for a `DataSource` made available by the container (JNDI support is not yet available for LWS, but may be enabled by a new add-on module later), or to a `DataSource` created by custom code, such as a servlet or listener.

Variable Name:	<code>javax.servlet.jsp.jstl.sql.dataSource</code>
Java Constant:	<code>Config.SQL_DATA_SOURCE</code>
Java Type:	<code>String</code> or <code>javax.sql.DataSource</code>
Set By:	<code><sql:setDataSource></code> , context parameter or custom code.
Used By:	<code><sql:query></code> , <code><sql:update></code> , and <code><sql:transaction></code>

The `maxRows` configuration setting can be set as a `String` value for a context parameter or as an `Integer` by custom code. It can be used to prevent run-away queries, since it sets a limit for how many rows are retrieved for a query result.

Variable Name:	<code>javax.servlet.jsp.jstl.sql.maxRows</code>
Java Constant:	<code>Config.SQL_MAX_ROWS</code>
Java Type:	<code>String</code> or <code>Integer</code>
Set By:	Context parameter or custom code.
Used By:	<code><sql:query></code>

I18N and Formatting Library Settings

Setting the fallback locale configuration setting provides a default locale to be used when the lookup of a locale based on user preferences (passed through the `Accept-Language` header value) fails to match an available locale. When `String` values are used to set these two variables it must be specified as a two-letter lowercase ISO-639 language code, optionally followed by a two-letter uppercase ISO-3166 country code, separated by a hyphen or an underscore character.

Variable Name:	<code>javax.servlet.jsp.jstl.fmt.fallbackLocale</code>
Java Constant:	<code>Config.FMT_FALLBACK_LOCALE</code>
Java Type:	<code>String</code> or <code>java.util.Locale</code>
Set By:	Context parameter or custom code.
Used By:	<code><fmt:bundle></code> , <code><fmt:setBundle></code> , <code><fmt:message></code> , <code><fmt:formatNumber></code> , <code><fmt:parseNumber></code> , <code><fmt:formatDate></code> and <code><fmt:parseDate></code>

Setting the locale configuration setting disables the lookup of a locale based on user preferences (passed through the `Accept-Language` header value). When `String` values are used to set these two variables it must be specified as a two-letter lowercase ISO-639 language code, optionally followed by a two-letter uppercase ISO-3166 country code, separated by a hyphen or an underscore character.

Variable Name:	<code>javax.servlet.jsp.jstl.fmt.locale</code>
Java Constant:	<code>Config.FMT_LOCALE</code>
Java Type:	<code>String</code> or <code>java.util.Locale</code>
Set By:	<code><fmt:setLocale></code> , context parameter or custom code.
Used By:	<code><fmt:bundle></code> , <code><fmt:setBundle></code> , <code><fmt:message></code> , <code><fmt:formatNumber></code> , <code><fmt:parseNumber></code> , <code><fmt:formatDate></code> and <code><fmt:parseDate></code>

The localization context setting can be set to a `String` value containing the name of the default resource bundle base name. The formatting actions then locate the locale specific version of this bundle. The `<fmt:setBundle>` sets the variable to an instance of the `LocalizationContext` class, which contains references to both a locale and a resource bundle for a locale.

Variable Name:	<code>javax.servlet.jsp.jstl.fmt.localizationContext</code>
Java Constant:	<code>Config.FMT_LOCALIZATION_CONTEXT</code>
Java Type:	<code>String</code> or <code>javax.servlet.jsp.jstl.fmt.LocalizationContext</code>
Set By:	<code><fmt:setBundle></code> , context parameter or custom code.
Used By:	<code><fmt:message></code> , <code><fmt:formatNumber></code> , <code><fmt:parseNumber></code> , <code><fmt:formatDate></code> and <code><fmt:parseDate></code>

The time zone configuration setting provides a default time zone for the JSTL actions formatting and parsing dates. `String` values for the time zone setting must be of the type defined for the `java.util.TimeZone` class: an abbreviation, a full name, or a GMT offset.

Variable Name:	<code>javax.servlet.jsp.jstl.fmt.timeZone</code>
Java Constant:	<code>Config.FMT_TIME_ZONE</code>
Java Type:	<code>String</code> or <code>java.util.TimeZone</code>

Set By:	<fmt:setTimeZone>, context parameter or custom code.
Used By:	<fmt:formatDate> and <fmt:parseDate>