Welcome
by Al Williams

Java & Transaction Processing
by Michael Trachtman
The Java Transaction Library (JTS) lets you create general-purpose transactions for e-commerce applications that work with any Java object.

Xlets & Mobile Communication Devices
by Eric Giguere
Xlets are like applets, but run on cellular phones and other mobile devices. Eric's simulator lets you run Xlets in a normal J2SE environment for development and debugging.

Document Diffing via Tokens
by Alain Trottier
TokenDiff, the diff tool Alain presented here, lets you compare by tokens (words), rather than lines. TokenDiff is built with an HTML front end and JSP diff engine on the back end.

Finding Web Services
by Aaron E. Walsh
Here's a program that finds nodes on peer-to-peer networks via UDDI, then uses JAX-RPC to deliver media content (such as movies and music) and calculate account balances.

Low Level I/O Control with Java
by Al Williams
Java's javax.comm library lets applications communicate with general-purpose embedded controllers. Al's javax.comm program uses the Mozilla project's Rhino library to provide a JavaScript interface, opening up a world of possibilities to write simple scripts that control real-world instrumentation and processes.

Java Bookshelf
by Mark Cyzyk
Mark takes a look at JSTL In Action, a new book by Shawn Bayern which examines the JSP Standard Tag Library.
Mark Cyzyk reviews Shawn Bayern’s JSTL In Action

The JSP Standard Tag Library (JSTL) provides a simple, tag-based programming language used for scripting web applications. Insofar as JSTL is Java-based—specifically, JSP-based—it runs on top of any JSP compliant “container” or Java application server.

The tags contained in this library encapsulate complex functionality into simple programming constructs. So, for instance, suppose you needed to programmatically retrieve a web page and store its content in a variable. Writing Java code to incorporate into a JSP page to perform this task would require several lines of code. Doing so using JSTL, however, is as simple as:

```<c:import url="http://someserver.com/somepage.html" var="somevariable" />```

This command connects to somewebsite.com and retrieves the somepage.html web page. It saves this content in a variable called, in this case, somevariable which, can then be used for further processing.

JSTL is actually comprised of four separate libraries:

- Core library.
- XML library.
- SQL library.
- Format library.

In his recently released book JSTL In Action, Shawn Bayern does a fine job of introducing these libraries and providing clear examples of each library construct in context.

The constructs of the Core library enable such common programming tasks as: setting the values of variables; outputting the value of a variable; tags for conditional logic and flow control; importing or including text from local or remote sources; and redirecting a browser to a different page.

The constructs of the XML library enable the reading, parsing, and searching of XML documents. The XML library also facilitates the building of conditional logic based on the contents of XPath search statements as well as iterative looping over XML nodes. Finally, it allows for XSLT transformations including the application of multiple stylesheets to a single document, resulting in a fine-grained degree of style control.

The constructs of the SQL library enable the setting up of datasources, and selection, insertion, updating, and deletion of records from such SQL-based datasources. Setting up a datasource is done using the `<sql:setDataSource>` tag:

```<sql:setDataSource driver="com.mysql.jdbc.Driver" url="jdbc:someDataBase:;" user="sa" password="somepassword" var="myDataSource" />
```

Once a datasource is set, it can either serve as the default datasource for a page or be referred to by its var value, if there are several separate datasources in use on a particular page. Usually, though, a default datasource is set for a page (or an entire application).

Once the datasource is set, it can be queried using the `<sql:query>` tag. For instance, the following example (borrowed from Web Development with JavaServer Pages, by Duane K. Fields and Mark A. Kolb, Manning Publishing, 2001), illustrates the benefits of using JSTL rather than direct JDBC code:

```<sql:query var="getRecords." sql="SELECT * FROM THING1" />
```

With JDBC/JSP, you’d need code such as this:

```<%= page import="java.sql.*" %>
<%=
Connection connection = null;
Statement statement = null;
ResultSet results = null;
try {
    Class.forName(oracle.jdbc.driver.OracleDriver);
    String url = jdbc:oracle:oci8@dbserver,;
    String id = request.getParameter(id);
    String query = "SELECT * FROM PRODUCTS_TABLE WHERE ITEM_ID = \" + id;
    connection = DriverManager.getConnection(url, "scott", "tiger");
    statement = connection.executeQuery(query);
    connection.close();
} catch (ClassNotFoundException e) {
    System.err.println("Could not load database driver!");
} catch (SQLException e) {
    System.err.println("Could not connect to the database!");
} finally {
    if (connection != null)
        connection.close();
} ```

JAVANET.COM
What takes a single line in JSTL requires several lines of syntax-laden code to accomplish in JDBC/JSP. Such is the power and simplicity of JSTL.

However, as Bayem stresses, the JSTL datasource connection does not provide a connection pool. So using JSTL datasources is only really viable for small applications—larger applications require a connection pool, encapsulated in an external Java library.

The Format library provides several functions used to format strings, numbers, currencies, and time/date stamps. These are all pretty straightforward utility functions. Of particular interest is the ability to override a web browser’s default locale settings. Using the <fmt:setLocale> tag, you could actually force a locale change from, say, French to Spanish if you so desired.

The rest of the book covers such things as how to perform common programming tasks in JSTL (for example, how to deal with check boxes, read dates, handle errors, and perform server-side validation of user input), as well as larger-scale examples of applications written in JSTL and other advanced topics.

Overall, the JSTL In Action provides an excellent introduction to JSTL. It is written with style and wit and is the most entertaining technical book I’ve ever read. It is certainly the most hilarious—Bayem’s examples frequently verge on the outrageous.

Mark is the web architect at Johns Hopkins University. He is also studying in the doctoral program in Information Systems at the University of Maryland, Baltimore County. He can be contacted at mcyzyk@jhu.edu.