Web Tools Platform:
J2EE Development the Eclipse Way

by Arthur Ryman

The story of WTP 1.0, its scope, design principles, architecture, ecosystem, and plans

NEED HEAD SHOT

Arthur Ryman is the leader of the Eclipse WTP Web Standard Tools sub-project. He is a software development manager and architect at the IBM Toronto Laboratory where he has spent the last 10 years working on J2EE development tools including VisualAge for Java, VisualAge for e-business, IBM XML and Web Services Development Environment, WebSphere Studio Application Developer, and Rational Application Developer. Arthur is a member of the W3C Web Services Description Working Group and an editor of the WSDL 2.0 Core Language Specification and Test Suite. He is an adjunct professor of computer science at York University, a senior member of the IEEE, and a member of the IBM Academy of Technology. He holds a PhD in mathematics from Oxford University. Arthur is a co-author of *Java Web Services Unleashed*, and is currently working on a new book with co-authors Naci Dai and Lawrence Mandel about WTP titled *Java Web Application Development with Eclipse*. *ryman@ca.ibm.com*

The Eclipse Open Source Integrated Development Environment (IDE) (see http://eclipse.org) is rapidly gaining popularity among Java developers primarily because of its excellent Java Development Tools (JDT) and its highly extensible plug-in architecture. Extensibility is, in fact, one of the defining characteristics of Eclipse. As the Eclipse home page says, "Eclipse is a kind of universal tool platform – an open extensible IDE for anything and nothing in particular." Although Eclipse is itself a Java application, all tools, including JDT, are on an equal footing in that they extend the Eclipse platform via well-defined extension points.

Of course, an infinitely extensible, but empty, platform might be interesting to tool vendors, but very boring for developers. Therefore, the initial version of Eclipse came with the JDT and the Plug-in Development Environment (PDE), both examples of how to extend the platform and very useful tools in their own right. JDT supported J2SE development while PDE supported Java-based Eclipse plug-in development. The combination of JDT and PDE fueled the creation of thousands of commercial and Open Source plug-ins for Eclipse, many of which supported J2EE development. For example, IBM released Eclipse-based commercial J2EE products, including WebSphere Studio Application Developer, and Rational Application Developer, while eteration, JBoss, Genuitec, Exadel, and Innoopract among others, released Open Source offerings. However, the profusion of J2EE plug-ins made it difficult for vendors to build on each other and for users to assemble an integrated suite of tools. For example, each J2EE toolset had its own way to support application servers.

As the popularity of Eclipse grew, it became apparent that the next logical step in its evolution was to add platform support for J2EE. This support would provide a common infrastructure for all J2EE plug-ins, with the goal of improving tool integration, reducing plug-in development expense, and simplifying the J2EE development experience for Eclipse users.

In June 2004, based on a proposal from IBM, the Eclipse Management Organization (EMO) agreed to create a new top-level project, the Web Tools Platform (WTP). However, it was believed that for WTP to be truly successful it needed a broad base of vendor support. A search began to engage additional vendors to partner with IBM. WTP was discussed in a BOF session at the first EclipseCon conference held in February 2004, and ObjectWeb agreed to lead the project creation effort. ObjectWeb assembled a set of vendors to join the project and agreed to co-lead the Project Management Committee (PMC). WTP was formally launched in June 2004 based on initial contributions from eteration, Lomboz, and IBM Rational Application Developer.

WTP got further industry endorsement earlier this year when BEA joined the project and announced plans to base a future version of WebLogic Workshop on it. BEA co-leads the PMC along with ObjectWeb. At this year's EclipseCon, Sybase announced the Data Tools Project (DTP), which will add to the data tools in WTP and create a platform layer dedicated to

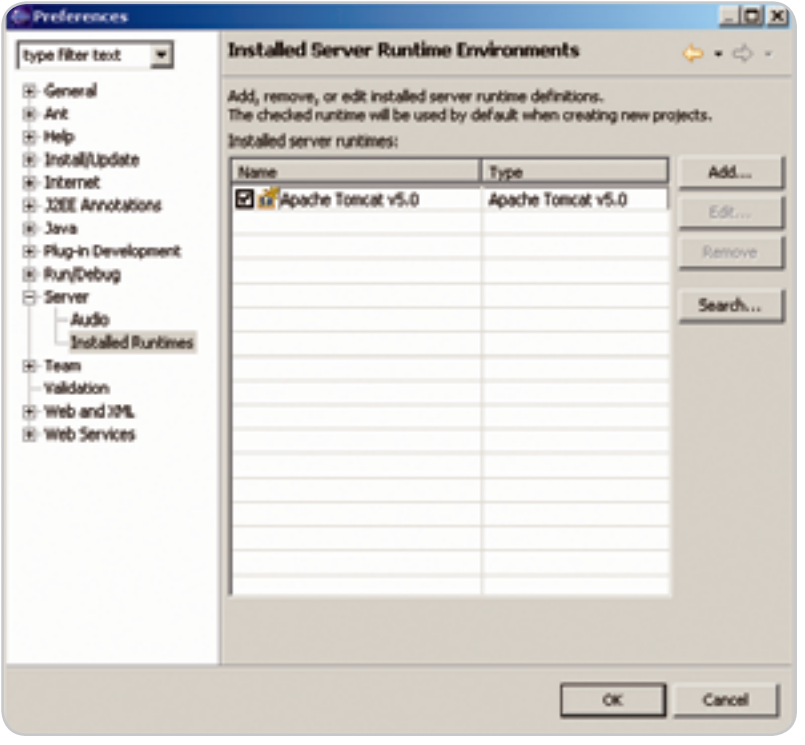

Figure 1 Server preferences page

database access. Oracle and Borland also announced Eclipse projects closely related to WTP. With major vendors such as IBM, BEA, Borland, Oracle, and Sybase all co-operating on a shared Open Source tool infrastructure, the center of gravity for J2EE tools has clearly shifted to Eclipse.

WTP 1.0 development is now well underway and has released a series of milestone drivers that can be downloaded from http://eclipse.org/webtools. The final release of WTP 1.0 is on track for a July 2005 delivery. The rest of this article gives you an overview of WTP, its scope, design principles, architecture, ecosystem, and plans.

A Quick Tour of WTP

One way to understand WTP is that it extends Eclipse along two dimensions, namely execution environments and artifact types. The execution environment dimension defines where code runs. Out-of-the-box, Eclipse lets you develop Java main programs that run in a command shell, applets that run in a Web browser, JUnit tests that run in a JUnit runner, and ANT tasks that run in ANT. WTP extends Eclipse by adding servers in general, and both J2EE and database servers in particular, as new execution environments. In general, you need to install an execution environment, configure it in Eclipse, and associate it with development artifacts that you want to run in it.

The development artifact dimension defines what developers create. Obviously, Eclipse majors in Java source code as a primary development artifact. However other artifacts, such as PDE plug-in manifests and Ant build scripts, are also supported. Each artifact type has associated with it builders, creation wizards, syntax-aware editors, validators, semantic search extensions, and refactoring support. Eclipse users expect editors to provide first-class programmer assistance such as code completion, syntax coloring, error markers, and quick fixes. WTP extends Eclipse with support for the large set of new artifact types encountered in J2EE development. These include HTML, CSS, JavaScript, XHTML, JSP, XML, XSD, WSDL, SQL, and all the J2EE deployment descriptors.

One of the key design goals of WTP is to extend Eclipse seamlessly to support these additional execution environments and artifact types. All of the functions that Eclipse users have come to expect from Java source code should "just work" for the new artifacts. For example, if I select a Java main program, I can Run or Debug it. The same should apply to a JSP. When I select it, the Run command should do something sensible. Specifically for a JSP I expect the Run command to somehow deploy my code into a J2EE server and launch a Web browser with the URL for my JSP. Similarly, the Debug command should run my J2EE server in debug mode and the standard Eclipse Debugger should let me step through my JSP source code. My JSP editor should provide code completion for both JSP tags and inlined Java scriptlets. Furthermore, I expect the code completion for Java scriptlets to work exactly like the code completion for Java source files. I don't want to learn new editing commands simply because I'm editing a new artifact type.

WTP 1.0 achieves many of these goals but there is much work to do to support J2EE fully. Consider the problem of refactoring a J2EE application. An operation as simple as renaming a Java class can have many consequences. If the renaming isn't fully rippled through the application, a runtime error can occur. For example, in addition to references from other Java classes, a Java class can be referenced by JSPs and deployment descriptors. All of these artifacts must be updated to reflect the new name. Suppose the Java class is deployed as a Web Service and that WSDL is generated from it. The WSDL may also need to be regenerated. First-class refactoring of J2EE applications will be an on-going focus for WTP.

Now let's create a JSP version of "Hello, world." If you'd like to follow along, you'll need to do some setup. Download and install the latest stable driver of WTP from the Web site mentioned above. WTP provides support for many popular commercial and Open Source J2EE servers but doesn't include the runtimes. So you also need to install a server on your machine. For purposes of illustration, I'll use Apache Tomcat 5.0.28, which you can obtain from http://jakarta.apache.org/tomcat/. Finally, you'll need a full JDK since JSPs require a Java compiler. I'm using Sun J2SDK 1.4.2_06.

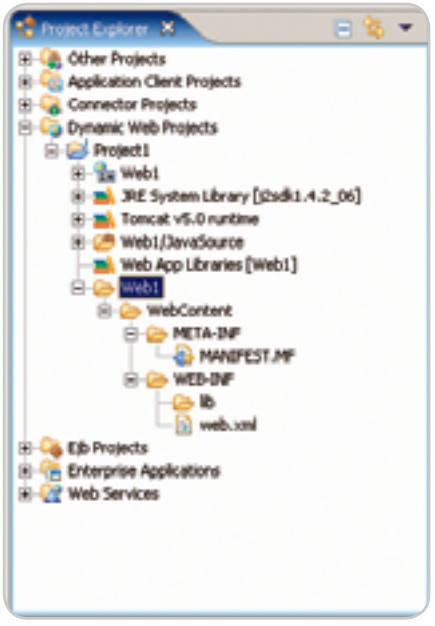

Figure 2 Project Explorer

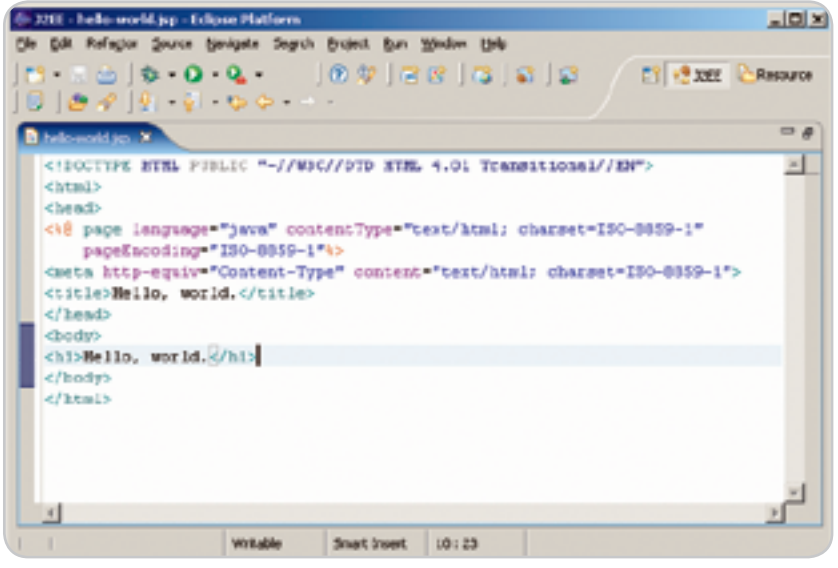

Figure 3 JSP editor with hello-world.jsp

WTP provides a Preference page for Servers. Open the Preference dialog and go to the Server page. Add your Tomcat 5.0 server and configure it to use your JDK (if you use a JRE then JSP compilation will fail). Figure 1 shows the Server Preference page.

Next, create a new Flexible Java Project named Project1 and a new J2EE Web module named Web1 in it. A Flexible Java Project is a J2EE project that can hold several J2EE modules. Figure 2 shows the J2EE Project Explorer after Project1 and Web1 have been created.

Now we're ready to create our JSP. Select the WebContent folder of the Web1 module and use the New File wizard to create a JSP named hello-world.jsp. The wizard fills in the skeleton of a JSP document and opens the file with the JSP editor. The JSP editor has full content assist for HTML and JSP tags, as well as Java scriptlets. Edit the file to say "Hello, world" and save it. Figure 3 shows the JSP editor.

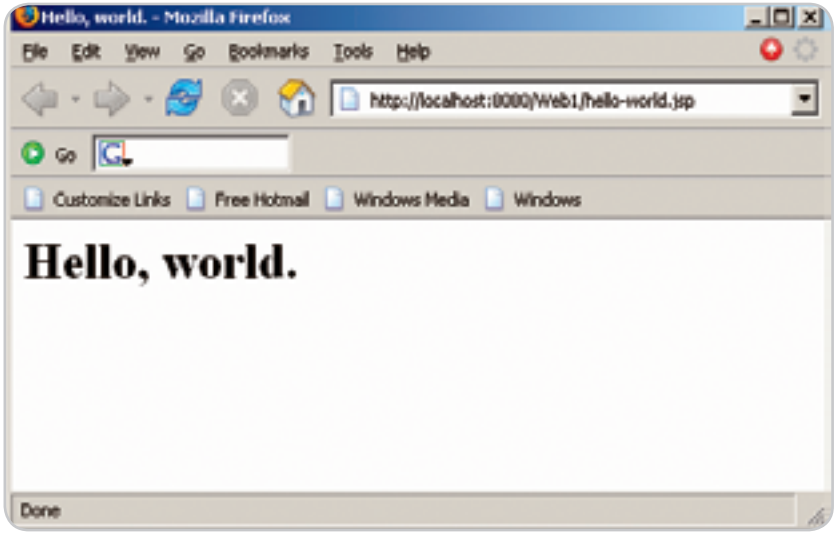

Figure 4 Web browser with hello-world.jsp

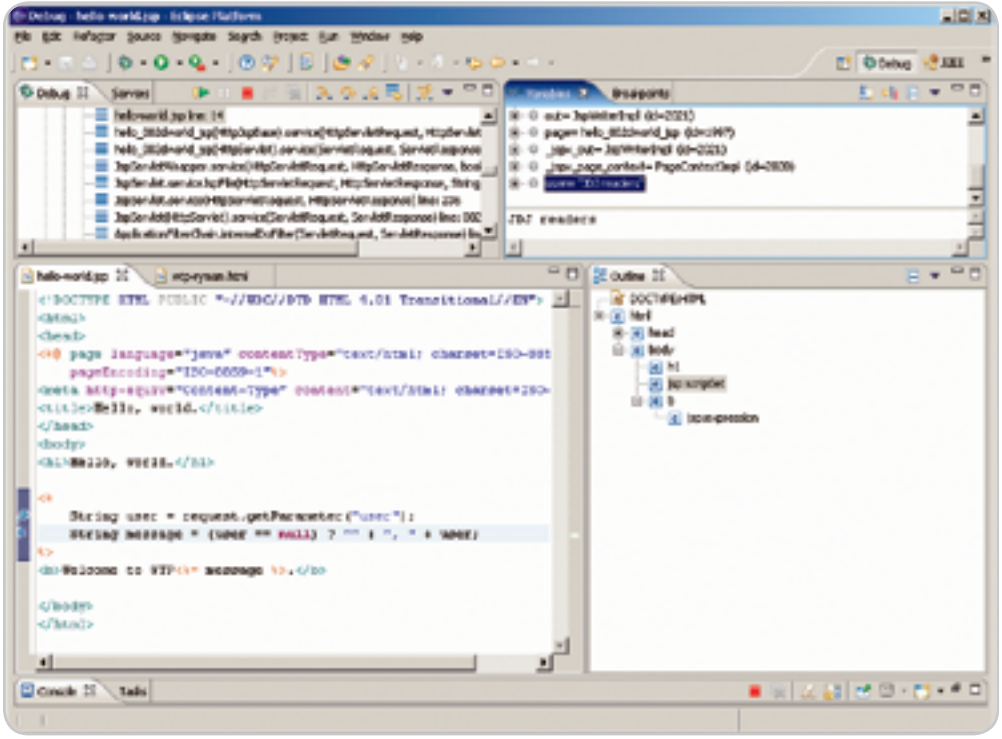

Figure 5 Debug perspective

Finally, we're ready to run the JSP. Select hello-world.jsp in the Project Navigator and the Run on Server command from the context menu. You'll be prompted to define the server to be used for the project since this is the first launch. Select the Tomcat server you previously defined and make it the project's default. The Web1 module will be added to the server configuration and the server will start. A Web browser will then be launched with the URL for hello-world.jsp. Figure 4 shows the Web browser with the Web page generated by hello-world.jsp.

Debugging JSPs is also simple. To demonstrate debugging, let's add a Java scriptlet to the JSP. The scriptlet will retrieve a query parameter named "user" and display a welcome message. Listing 1 shows the modified JSP that includes the Java scriptlet.

Set a breakpoint on the line that begins String user = by double-clicking in the left margin. Select the file hello-world.jsp in the Project Explorer and invoke the Debug on Server command from the context menu. The server will restart in debug mode and the Debug Perspective will open with execution halted at the breakpoint. Figure 5 shows the Debug perspective when the JSP is passed the query parameter user=JDJ readers on the URL.

You can now step through Java scriptlets and explore Java variables as usual. In Figure 5, the variable user has been selected in the Variables view that shows its current value JDJ readers. Click the Resume icon to finish processing the JSP. Figure 6 shows the Web browser displaying the resulting Web page.

WTP 1.0 Features

The preceding Quick Tour shows a small cross-section of the features available in WTP 1.0. The full set of features in WTP 1.0 includes server, Web, XML, Web Service, J2EE, and data tools. I will now briefly describe these. Consult the WTP Web site for more details.

The server tools let you define and control servers. Servers can be associated with projects, and can be started, stopped, started in debug mode, and controlled in other ways. Projects can be deployed to servers, or servers can be configured to access your Eclipse

workspace content directly. WTP includes server support for Apache Tomcat, Apache Geronimo, and JBoss, as well as many other popular commercial and Open Source J2EE application servers. The server tools include an extension point so that new server types can be easily supported. You can add a new server type by providing either a simple XML configuration file or a Java plug-in.

The Web tools let you create static Web pages based on HTML, XHTML, CSS, and JavaScript. The Web tools include source editors that are based on the WTP Structured Source Editor (SSE) Framework. WTP Web editors provide content assist, syntax highlighting, validation, and other standard Eclipse editor functions. The Web tools also include an embedded Web browser and a TCP/IP monitor that's very handy for debugging HTTP traffic.

The XML tools include source editors for XML, DTD, and XSD that are based on the SSE Framework. Besides source editing, graphical editing is also provided for XSD. The XML tools also include code generators for creating XML instance documents from DTD or XSD.

The Web Service tools include a WSDL editor, a Web Service Explorer, a Web Services Wizard, and WS-I Test Tools. The WSDL editor includes an SSE-based source editor and a graphical editor. XSD editing is seamlessly integrated with WSDL editing. The Web Service Explorer lets you search and publish to UDDI registries, and dynamically test WSDL-based Web Services. The Web Service Wizard ties together the full development lifecycle. It lets you deploy Java classes as Web Services, generate server and client code from WSDL, and generate test clients, as well as being integrated with the Web Service Explorer for publishing and discovery. The Web Service Interoperability (WS-I) Test Tools let you validate WSDL and SOAP for compliance with the WS-I profiles.

The J2EE tools let you create J2EE projects and artifacts like JSPs, servlets, and EJBs, as well as the J2EE deployment descriptors, and deploy these to app servers. The J2EE tools have an SSE-based JSP source editor and a J2EE Project Navigator that displays J2EE components as objects. This provides a higher-level view of your project resources, for example, by displaying all the files related to an EJB as a single EJB object.

The data tools include support for connecting to JDBC-based databases such as Cloudscape, Derby, and other commercial and Open Source databases, and exploring their tables. The data tools also include an SQL source editor that lets you easily execute SQL statements and view the results.

The WTP Noosphere

In his essay "Homesteading the Noosphere," Eric Raymond likened Open Source developers to homesteaders who stake out their turf in the sphere of ideas. I will therefore describe the turf that WTP has staked out in the Eclipse noosphere.

WTP components are organized along the lines of open standards. WTP classifies the world of standards along two dimensions – technology and formality.

Figure 6 Web browser with modified hello-world.jsp

The technology dimension ranges from neutral standards on one extreme to J2EE standards on the other. Technology neutral standards form the foundation of the Web. In fact, the Web has succeeded because it's defined in terms of formats (such as HTML and XML) and protocols (such as HTTP) that specify how systems interact, but don't specify how systems are implemented. This neutrality has allowed vendors to choose the most appropriate implementation technology and compete on the basis of the quality of their implementations. On the other hand, J2EE standards specify application portability rules for J2EE implementations. Both kinds of standard are essential for the viability of the Web.

The formality dimensions define how the standards are created. At one extreme we have the de jure standards bodies such as ISO and IEEE and at the other we have technologies that aren't associated with any formal standards definition organization, but have become de facto standards because of their popularity. Organizations such as W3C, OASIS, and JCP, which define the standards relevant to WTP, have formal processes and are near the de jure end of the spectrum. Popular Open Source technologies such as Struts and Hibernate are at the de facto end.

Figure 7 shows the world of standards classified along the technology and formality dimensions. The turf of WTP is, in principle, all the important standards that are relevant to Web application development. However, the charter of WTP focuses on the standards that are formally defined by recognized standards-definition organizations, i.e., those that cluster towards the de jure end of the spectrum. WTP consists of two sub-projects, Web Standard Tools (WST) and J2EE Standard Tools (JST) that cover the formally defined Web and J2EE standards.

The reasoning behind this scope is that WTP aims to be a platform that many vendors can build on. The formal standards form the building blocks that most vendors want. Support for this base layer of standards is, in a sense, the "table stakes" of any tool. Vendors can cooperate on this

base layer and produce high-quality common components while sharing the development expense. Conversely, by not including the de facto standards, WTP leaves room for vendors to innovate and differentiate. For Open Source to succeed contributors must have a way to generate a profit otherwise they won't be able to continue contributing. We hope that this design will yield an excellent set of core Open Source J2EE tools for users, and a solid platform that supports a thriving aftermarket of extenders.

The standards arena is very active and as existing standards are revised and new standards defined WTP will support them based on their market relevance. There may also be a migration of de facto standards to the de jure quadrants. WTP's charter may expand in the future to include new sub-projects. However, immediately, WTP consists of the WST and JST sub-projects.

The WTP Ecosystem

WTP has the dual goals of providing both tools for the developer community and a platform for tool vendors to extend. Satisfying the needs of vendors requires that WTP define a set of platform APIs. The significance of a platform API is that it will be preserved in future releases. This means that a plug-in that runs in WTP 1.0 will also run – without recompilation – in future versions of WTP. The stability of platform APIs is key to vendor adoption. Clearly if WTP changed its APIs from release to release, vendors would expend significant effort reacting to the changes, and this would slow the rate at which users and vendors move to new versions of the platform.

WTP relies heavily on the user community for testing, bug reports, and enhancement requests, and the development of the user community is one of our main focuses this year. The WTP Web site has tutorials, articles, presentations, and event information. WTP will be well represented on the conference circuit this year. Look for upcoming WTP presentations at events such as EclipseWorld, JavaOne, and the Colorado Software Summit. There are also a couple of WTP books in the works. A thriving user community is a magnet for vendors. As the WTP user community grows so will the number of tools built on it.

Finally, WTP has a role to play in education. Since WTP is free Open Source and supports industry standards, it's an ideal learning tool for the coming generation of J2EE developers. I hope to see universities, community colleges, and even high schools use it for teaching.

The WTP contributor community is drawn from both vendors and users. There are many ways to contribute. You can start by downloading WTP, kicking the tires, and telling your friends about it. If you find a problem or have an idea, open a Bugzilla report. Monitor the newsgroup, and share your solutions to problems with others. If you can write, submit a tutorial or contribute to the online Help system. If you have fixed a problem, submit a patch. If you have time to work on WTP, check Bugzilla for open problems or look at the WTP Help Wanted page. And after you have established a track record of valuable contributions, you can be voted in as a committer.

What's Next?

WTP 1.0 is scheduled for release in July 2005. We are planning to follow that with WTP 1.1 later in the year. The focus of WTP 1.1 will be on the further development of platform APIs to enable the first wave of products based on WTP. Following that, WTP 2.0 will be released with Eclipse 3.2 in 2006. Candidate items for WTP 2.0 include support for revisions of major specifications such as J2EE 1.5, SOAP 1.2, and WSDL 2.0, as well as new JSRs and Web Service specifications.

We also expect the shape of WTP to change as new projects emerge and mature at Eclipse. New vendors are joining Eclipse and projects are being created at a rapid clip. For example, the data tools in WTP will move into a new Data Tools Project. Technology projects such as those proposed for EJB 3.0 and JSF will likely move into WTP as they mature.

A Final Word

Like all Open Source projects, the success of WTP depends on the contributions of an enthusiastic community. The project is still in its formative stage and there's much work to do. The project needs users, testers, writers, developers, speakers, trainers, mentors, evangelists, extenders, distributors, and leaders. If you are interested in J2EE development, then please consider this article as your formal invitation to join the WTP community.

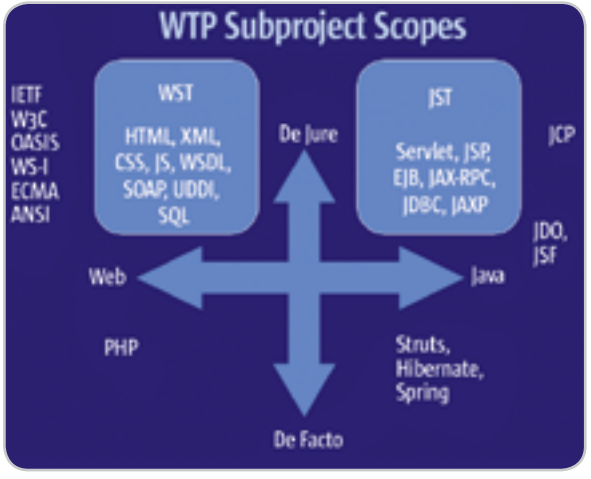

Figure 7 The scope of WTP

Listing 1: Hello-world.jsp with Java scriptlet

```
        <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Hello, world.</title>
</head>
<body>
<h1>Hello, world.</h1>

<%
 String user = request.getParameter("user");
 String message = (user == null) ? "" : ", " + user;
%>
<b>Welcome to WTP<%= message %>.</b>

</body>
</html>
```