Eclipse Board Review of WTP:

An Informal Report

September, 2008

David Williams

Table of Contents

1 Introduction

This document is to provide an informal, high level review of the Web Tools Platform (WTP) project, and was created at the request of the Eclipse Board as preparation for a phone conference on September 17, 2008, and will (likely) become a regular yearly activity to improve communication and awareness.

The Eclipse Web Tools Platform project was originally proposed in 2004 by ObjectWeb, IBM and others. The Eclipse Foundation creation review was in June, 2004 with full time development since October, 2004. The original code contributions were from IBM and Eteration ("ObjectWeb Lomboz"). Since then several other large contributors have joined the effort, including SAS, BEA, Oracle and SAP, and others, and several new subprojects formed, such as Dali and JSF.

More information can be found at our WTP web page and Wiki site. In addition, our most recent release review contains much more formal detail, if desired.

1.1 Releases

- WTP 0.7 July, 2005 and subsequent 0.7.1
- WTP 1.0 December 2005 and subsequent 1.0.1, 1.0.2, 1.0.3
- WTP 1.5 June, 2006 and subsequent 1.5.1, 1.5.2, 1.5.3, 1.5.4, 1.5.5
- WTP 2.0 June, 2007 and subsequent 2.0.1, 2.0.2, 2.0.3
- WTP 3.0 June, 2008 and subsequent 3.0.1, 3.02, and 3.0.3 (planned)

1.2 Project Organization

In the original charter, WTP was organized as two sub-projects (WST and JST) with some being added later (JSF, Dali, and ATF (incubating) but was reorganized into more sub-projects primarily to help emphasize more of a team-oriented focus, instead of an architecture orientation, and a new WTP Incubator Project was added. Below is the list of current projects and project leads as of September, 2008.

We have a regularly occurring WTP-wide status meeting (every week), and these sub-projects occasionally schedule their own meetings as they need them and document those meetings on our WTP Wiki.

Project	Lead
Common: tools and infrastructure not directly related to web tools, but required by Web Tools Platform	Konstantin Komissarchik, Oracle
Dali (JPA Tools): infrastructure and tools for JPA applications	Neil Hauge, Oracle
EJB Tools: EJB creation wizards, preferences, future annotation tools	Kaloyan Raev, SAP
Java EE Tools: Common Project Infrastructure, JEE models, preferences, classpath model, publish api, refactoring	Chuck Bridgham, IBM
JSF Tools: infrastructure and tools for Java Server Faces.	Raghu Srinivasan, Oracle
Server Tools: tools and infrastructure to define and interact with servers.	Tim Deboer, IBM
Source Editing: xml, dtd, xsd (and sse infrastructure) html, css, javascript, jsp	Nitin Dahyabhai, IBM
Web Services:: Web services wizards and frameworks, Axis1 & Axis2 support, Web Services Explorer, WSDL Editor	Kathy Chan, IBM
Release Engineering: contains the code and scripts to do builds, various tests, API scans, etc.	David Williams, IBM
WTP Incubator: a general purpose incubation project other WTP Projects to use when incubation is desired.	David Williams, IBM
ATF (incubating): infrastructure and tools for AJAX	Philippe Ombredanne, nexB
Datatools (RDB): tools for working with databases. Primarily moved to DTP, but this quiescent sub-project of WTP occasionally does 1.5 maintenance	Der Ping Chou, IBM

1.3 PMC Organization

Our Project Management Committee, as of September, 2008, is made up of 6 members from several companies and we are all quite active in WTP, and have a long history of weekly PMC meetings.

Each member has a WTP-wide management role, in addition to whatever project-specific roles they have. In the execution of their tasks, within these roles, the PMC members will form groups, organize meetings, create and execute plans to accomplish their goals. In other words, they don't do all the work ... they just manage it!

Member	Role
David Williams, IBM	PMC Lead, and Planning Role
Tim Deboer, IBM	Architecture
Neil Hauge, Oracle	Quality
Kaloyan Raev, SAP	User Experience
Raghu Srinivasan, Oracle	Requirements
Naci Dai, Eteration	Education

2 Review of project scope and charter

WTP, as our original charter states, is still "... dedicated to providing a generic, extensible, standards-based [and vendor neutral] tool platform for producing Web-centric technologies" The project operates within its intended charter and scope; there have been no changes in the substantive areas of WTP or intent. WTP is still primarily "Java oriented" (Java EE, JSF, JPA), but non-Java language tools (such as HTML, CSS) are also strong and some even growing in activity, such as Javascript and XML.

Since the original charter, there has been a number of additions (Dali, JSF, ATF, WTP Incubator) and a major refactoring of the original two projects. Each change was duly documented, reviewed, and approved but this project evolution has resulted in very fragmented documentation.. An updated charter, that would improve the descriptive value, could be developed with the next major revision or project addition. Another pragmatic suggestion is that as moves and re-organizations take place, each of those documents should be better tied to (or linked from) a central document about the project, so the history stays with the current state.

There are two areas we should discuss in more detail: "vendor neutrality" and "standards based". While we certainly agree with the Eclipse-wide concept of "vendor neutrality", how does that intersect with Eclipse Projects themselves? Especially with the addition of the Eclipse Runtime Project? For example, if someone contributed a Web Application server to the Eclipse Runtime Project could we "ship it" with our Java EE IDE package? Could we favor it in writing our exemplary tools? I'm not asking for "yes or no" responses for these hypothetical cases ... just wondering if the Board has some principle to guide such issues?

Second, what is the reason or history behind the restriction of WTP working only on "standards based" technology? Is that still relevant or required? We are getting into some areas where that is being stretched (e.g. ATF, JSF 2.0) and since clearly things like Struts or Spring tools could be done in some other Eclipse Project (such as Technology) and then presumably packaged up in the cross-project Java EE IDE, then why have WTP specifically limited to not hosting the work? While this is not a limiting factor right now, we think it might be in the next year or two so we'd like to understand why this was put in place and remove it if there is no current reason for it, or modify it so it is meaningful in the current Eclipse organization.

3 Review of progress, strategy and plans

We have had a long history of steady, predictable releases and plan to keep doing that. We do plan to always participate in yearly simultaneous releases, as long as there are any, but subprojects are free to have additional releases out of that yearly cycle – for example, Dali plans a release 2.1 this December, 2008.

For the most part, our progress and plans are based on the progress of standards, with Java EE 6 and JSF 2.0 being some of the major new ones coming up (the planning for which is still in progress).

Our WTP Incubator Project has successfully attracted a group of independents that are doing great work in the area of XSL and other XML technologies ... at least some of which are expected to graduate and have formal releases in 2009.

Self assessment

This self assessment is very subjective. In most cases, I deliberately tried to think of both positive and negative aspects in each category and hope that gives some gist of the state of our overall project, but it doesn't necessarily "measure" each area.

3.1 As Eclipse Open Source Project

3.1.1 Transparency: Making our internal processes apparent to others

I think we are nearly as transparent as we can be. There is always some "corporate discussions" that go on, specific to the business interests and priorities and plans of the primary sponsoring corporations ... which are confidential ... so there are sometimes periods of "quietness" while these discussions occur ... but we say that's what we're doing, so not much more to say there.

3.1.2 Openness: Accepting input and contributions from others

I think we do a fair job here. We do pay attention to bug votes (even increased the maximum to 20 based on a newsgroup suggestion). We have at times not reviewed patches in a timely manner ... such that we started special "reviews" to make sure we made progress on them. But I think the reason for this isn't so much a symptom of not being open, just that it takes a lot of effort, and the return is relatively low (my intuition is we might be able to use only 25% or so).

3.1.3 Meritocracy: Responsibility (and power) follows contributions

We do a good job in all the obvious ways; contributors are voted in as committers, the PMC keeps an eye on likely candidates and discusses with Project Leads if apparent candidates are not being proposed in a timely fashion. There is still, rarely, some hints of an entitlement attitude (the opposite of meritocracy in this context) but I think in most cases this is not so much having the wrong attitude as it is that some areas of our code are hard, complex, and takes a major investment to get started.

3.1.4 Diversity: many interests participate towards common goals

Excellent. We have several large corporations with major contributions (IBM, Oracle, and SAP) and several smaller ones and some "independents". There have been some known cases of one group "taking over" when another group could not continue making their contribution ... one of the fruits of diversity. There is not always great diversity within each sub-project (some have less diversity than others) but each has some diversity, and, in my opinion, some homogeneity in sub-projects is important (so responsibilities are clearer within that sub-project).

3.1.5 Compliance with Eclipse prime directives

3.1.5.1 Exemplary, extensible tools

We do surprisingly well here (given the “API” case, below): there are many adopters building on and extending WTP in many interesting ways. And the tools, direct from WTP, while not state-of-the-art tools that can be purchased, have many satisfied users.

3.1.5.2 API

In my humble opinion, we do poorly here. We certainly do have API and extension points and we certainly do add to them every release, but, from my viewpoint, our committers are too cautious in declaring API. There is a tendency for them to want to wait until they know it is just right, spanning one or two releases before declaring official API. Also, sponsors often don't see the return-on-investment in doing the extra work to make a high quality API (and ... it is expense!). This situation is partially due to our beginnings (starting with a large existing code base, instead of starting from scratch). We have taken steps to guard adopters investments when they have had to use non-API, by creating some specific non-API policies.

3.2 End user community and adoption

End user community and adoption is very strong. One of the most popular, frequently downloaded packages, and very active newsgroups and mailing lists. One sign of success, to me, is that our newsgroup is very much “user supported” ... that is, users helping users. While our commiters do participate, some people have made the observation to me that they could do better. I personally think they do quite well, considering all the conflicting priorities they work under.

3.3 Commercial community and adoption

Too many to keep track of.

3.4 Compliance with the Roadmap

Roadmap? What roadmap? Just kidding. We do pay attention to it, and try to categorize our work in terms of the roadmap, but I think the Eclipse Roadmap is generally thought of (through out Eclipse) to be broad enough that any work could fit in somewhere! But, we are aware of our short comings in the area of ease-of-use for new, casual users and do invest in making progress in this area, though it's slow going.

4 Notes on each subproject

Much of time WTP acts as one-big-project but there are differences between the subprojects (in both history, plans, and assessments) so the following list describes each subproject a tiny bit more and highlights some specific strengths and weaknesses.

4.1 Common Tools

Tools and infrastructure not directly related to web tools, but required by Web Tools Platform.

Some very useful APIs and frameworks (e.g. project facets, validation). There's also some packages that (in hindsight) do not belong and could use some refactoring to more specific sub-projects. If Eclipse ever has a common components project some of this subproject could move there.

4.2 Dali (JPA Tools)

Infrastructure and tools for JPA applications.

Primarily one-company contributions, but many-companies adopt (and test, and open bugs, and make requirements). Close affinity to Eclipselink, but can be used with other implementations of JPA runtimes.

4.3 EJB Tools

EJB creation wizards, preferences, future annotation tools

In practice, this is currently highly intertwined with the JEE Tools subproject, but the hope is it can allow some future specialization and divisions of labor, code, and architecture.

4.4 Java EE Tools

Common Project Infrastructure, JEE models, preferences, classpath model, publish api, refactoring

As a team, this component is the core of WTP. It is a difficult area of code, since they support many levels of Java EE, and have tried to “change architecture” over the years (so that different levels of Java EE specification can be better plugged in to the frameworks.

4.5 JSF Tools

Infrastructure and tools for Java Server Faces.

Very strong JSF expertise. Good adoption. Probably one of the most innovative subprojects, having good visual editors for JSF and doing forward looking (incubating) work with Facelets.

4.6 Server Tools

Tools and infrastructure to define and interact with servers.

The most mature of all the WTP subprojects. Very mature API and a great many adopters.

4.7 Source Editing

xml, dtd, xsd (and sse infrastructure) html, css, javascript, jsp.

One of the most important subprojects in all of Eclipse (I'm biased, though, since I was the former lead) But, seriously, it is one of the most extended areas of WTP. Unfortunately, due to so many languages to support, and so few people, the support for these languages may never reach parity with the JDT's Java source editors. But, the areas of XML and Javascript (JSDT) are still quite popular and attracting new committers.

4.8 Web Services:

Web services wizards and frameworks, Axis1 & Axis2 support, Web Services Explorer, WSDL Editor.

A subproject well known for its end user tools and editors in WTP.

4.9 Release Engineering

Contains the code and scripts to do builds, various tests, API scans, etc.Mostly a “technicality” just to keep track of who can do what to our builds ... but, we do have some original code and a pretty good build system.

4.10 WTP Incubator

A general purpose incubation project other WTP Projects to use when incubation is desired.

One of the bright spots of the past year. A very dedicated group of XML specialist have been adding XSL tools that are expected to graduate and release concurrently next year. Facelets (especially the future JSF 2.0)is another active area here, and a few other recent proposals.

4.11 ATF (incubating)

Infrastructure and tools for AJAX

An area with great potential, but not many committers able to invest significant time. We do still expect some activity later this year, and if all goes well, graduation and formal release can still be achieved next year.

4.12 Datatools (RDB):

Tools for working with databases. Primarily moved to DTP, but this quiescent sub-project of WTP occasionally does 1.5 maintenance Pure history. But, used in some major products, so will require maintenance for years.

5 Board Assistance and noteworthy items

I have combined the last two requested review categories because we don't have any specific requests of the board at this time. But, some noteworthy items might turn into requests in the future. First, let me say I think the board had done a good job in solving the IP Backlog problem (such that it is no longer a problem, in my opinion). And, the Eclipse Foundation does a great job in providing infrastructure for builds, bug tracking,, etc. If there were some areas to note at all:

- As a world-wide, multi-corporation group, we have a hard time, sometimes, collaborating in real time. I've heard there is more the Foundation could do here ... in providing a universal Instant Message (IM) chat client/server (see bug 126089) but I don't actually know enough about it technically to be more specific. I do know, that what ever Eclipse infrastructure does, there will be issues of corporate policy permissions and corporate firewall rules to overcome. So, just note, this is an area where we might have more specific requests in the future.
- One goal I have this year is to investigate and encourage using more “Java” on Eclipse.org itself, both to showcase our tools and create example applications, but also (and mostly) to provide more opportunity for “self hosting,” so to speak. While I think some of the required infrastructure (e.g. Tomcat servers) are available, we may want to discuss expansions in this area (e.g. could Eclipselink be installed and run on Eclipse?) Again, we are not far enough along in our planning or discussions of this effort to have hit any roadblocks, but, as the year progresses, I'd appreciate the Boards awareness of this and helping us along the way, where possible.
- We do have one specific request. Can we, as a PMC, provide some input in to the next Evans (or Biz Media?) marketing survey that is commissioned? We'd like to see more specific questions about Web Tools with respect to end-user satisfaction and adopter satisfaction. It may be as simple as listing some of our sub-projects individually, but may make more sense to respondents if we listed some areas of technology, such as JEE, JSP, JSF, JPA, XML, etc.