Continuation Review for the Eclipse ATF (Ajax Tools Framework) Project By Philippe Ombredanne, Robert Goodman, David Williams

February 13, 2008

Executive Summary

The Eclipse ATF (Ajax Tools Framework) started in mid-2006 and has been incubating since in the WTP project.

Why are we doing a continuation review?
New project leadership and request for continued incubation status are motivating this review.

What has been accomplished?
ATF key contributions:

- Embedded Mozilla/Firefox aka Xulrunner (now part of SWT),
- Advanced JavaScript editing (now part of WTP sources editors)
- HTTP server adapter (now part of WTP)
- Browser based tools for Ajax and JavaScript
- Ajax and JavaScript in-browser debugger
- Ajax library provisioning
- Mozilla/Eclipse collaboration

Key adopters gained (more are underway)
Frequent milestones drops
Community outreach and involvement

What is the plan going forward?
ATF will continue incubating until Q1 2009 when it would officially exit incubation and integrate WTP as a standard component. Until then the project will focus on new features, stabilization, tests, delivering release grade milestones, and working towards the WTP integration and schedule synchronization with at least quarterly milestones drops.

Why should ATF continue ?
Ajax is hot and becoming a mainline technology (see JavaScript Dominates EMEA Development and JavaScript Now Outstrips Java),
ATF is the only Eclipse project that offers IDE-integrated Ajax tooling,
ATF is incubating in WTP, supports the WTP projects technology and goals and offers WTP-integrated support for Ajax technologies,
ATF has made key technology contributions adopted Eclipse-wide,
ATF commercial and community adoption is gaining ground,
Beyond the original committer team, new committers are joining the project, ensuring its long term viability,

Therefore:

For Eclipse and WTP to continue offering quality open source Ajax tooling to a growing population of Ajax developers and adopters, ATF should continue as an Eclipse project and integrate as a WTP component when exiting incubation.

Table of Contents

Introduction/Motivation/History

According to the Eclipse Development Process, a Continuation Review ...

"is to verify that a Proposal or Project continues to be a viable effort and a credit to Eclipse. The Project team will be expected to explain the recent technical progress and to demonstrate sufficient adopter, developer, and user support for the Project. The goal of the Continuation Review is to avoid having inactive projects looking promising but never actually delivering extensible frameworks and exemplary tools to the ecosystem."

Therefore, the purpose of this continuation review is :

- to review to accomplishments of the ATF project to date,
- and present the long terms plans for the project in terms of incubation, new features, committer-ship, communication and community.

This review is timely since the ATF Project has been in incubation for a while (about a year and a half) and in addition it happens to be going through a change in Project Leadership.

The ATF Project was proposed in January 2006 by Craig Becker (IBM) and was accepted after its creation review March 8th, 2006. Since then, it has been led by Robert Goodman. Bob has been asked recently by his employer (IBM) to consider new work assignments, and has been searching for a new project leader, and has found Philippe Ombredanne who is willing, capable and has plans to take the project to a Release. Philippe has been an ATF committer since early 2007.

ATF has had many successes to date such as:

- the introduction of the Embedded Mozilla XULRunner Browser support on multiple platforms which is now part of the Eclipse platform in SWT, helping close one of the oldest Bugzilla enhancement request,
- a standard basic HTTP Server adapter with is now part of WTP Server Tools,
- the JavaScript Development Tools (JSDT) which is now part to the Source Editing component of WTP,
- the develop browser based tooling (JavaScript Debugger, DOM Inspector, etc) which laid the foundation for future browser based tooling such as a WYSIWYG editor.
- being a force within Eclipse to drive changes for WTP to provide a Web Development Environment,
- fostering an active collaboration between two open source communities (Eclipse and Mozilla),
- serving as a force in the creation of the Open Ajax Alliance and taking part in the IDE work group.

Many of these successes have had the effect of transferring some significant components or technologies out of the ATF Incubating project into other Eclipse projects where they have been released.

ATF features today

The AJAX Tools Framework (ATF) is an Integrated Development Environment (IDE) for AJAX and DHTML developers. ATF provides tooling that allows a user to develop, debug and deploy Ajax/DHTML applications and a framework on which adopters can build advanced and technology specific tools. The functionality in ATF breaks down into three main areas.

1. Browser Tooling

The Browser tooling allows a developer to inspect and manipulate their DHTML application in the live browser. The Browser tooling consists of one editor and several views:
The **Mozilla Browser Editor** allows the embedded Mozilla browser to show up in the Eclipse editor area. The Mozilla browser editor communicates with other parts of the browser tooling.

The **DOM Inspector** shows the DOM tree rendered by the Mozilla Browser Editor. It is live, meaning that it will dynamically change to reflect changes within the browser. The DOM inspector also allows attributes of a selected DOM element to be modified and immediately see the effects in the browser.

The **Browser Console** shows all browser (i.e JavaScript, CSS) errors, warnings, and logging messages that occur at runtime.

The **Request Monitor View** is used to observe request/response information for HTTP calls.

The **JavaScript Evaluation View** enables the developer to explore and interact with the web application by evaluating JavaScript expressions.

The **DOM Source View** displays the HTML source of the selected DOM node. The source can be edited, validated, and updated back to the browser page's DOM.

The **CSS View** shows all style rules that are currently applied to a selected DOM node.

The **DOM Watcher View** is used to track DOM Events that target a given DOM Node.

The **DOM Compare View** allows DOM nodes to be compared based on DOM attributes, CSS, and child nodes.

2. JavaScript Debugger

The JavaScript debugger allows the developer to debug their AJAX/DHTML application. The JavaScript debugger allows a developer to set breakpoints, inspect the call stack, variables, etc. of their application.

3. Project Support for JavaScript and Ajax runtime libraries

ATF provides a Generic AJAX Runtime support will allow any Ajax Runtime to be defined to ATF and added to a project. ATF's project support allows a developer to deploy their application to HTTP and J2EE server for testing.

Some lessons learned from first year and a half of ATF

Web Developers have found Eclipse complicated and difficult to learn.

The base Eclipse with WTP provides features and functions for a wide range of developers focusing primarily on Java and JEE aspects of Development. In working with Web Developer many have said that there were too many options unrelated to Web Development. Since they were not Java/JEE developers they didn't know which options to use. One of the comment we heard was that there are twenty different project types that can be created, which one should a Web Developer use? In Eclipse 3.3 and WTP 2.0 a number of changes have been made that allows a Web Developer platform to be built. As Eclipse and WTP moves forward more thought should be given to needs of the Web Developer.

There are no real standards for Ajax runtimes.

Ajax runtimes are a essentially a collection of JavaScript libraries (with some interspersed DHTML or XML) and there are no standards for how the libraries are created, built, and the functionality provided. This has made difficult to provide Ajax specific tooling without picking certain libraries to support. The Open Ajax Alliance is attempting to define standards in this area. Until these standards are defined, accepted and adopted it will be difficult for ATF to provide Ajax specific tooling without picking certain libraries.

Ajax is a growing and highly evolving community.

Things change quickly in the community. There are a lot of different Ajax runtimes and they drop new versions of their runtime every three to four months, some even more often. Also the Ajax runtimes seem to go in and out of favor at about the same rate. With things changing as often in the Ajax community, it has been impossible to get through the Eclipse IP process review before the library changed. This has been one of the reason why ATF went to a Generic Ajax support and dropped support for specific libraries. In the process of moving to a Generic Ajax support the ATF team had to drop functionality like snippets and templates for certain Ajax libraries.

It is hard to get committers to join a project.

The ATF team have had a number of people wanting to contribute to the project, but in the process of trying to bring them aboard they lost interest. One of the biggest impediment seem to be that fact that individual committers need to have their company approve their participation. ATF has a number of adopters but they also didn't want to commit significant resources to the project which proved to be frustrating to the team. Potentially some help/direction for new projects on how to bring new committers on board may make this go smoother in the future.

This leaves us at a good point to step back, and consider the future direction of ATF, where we think it should go, and where it should focus based on community and adopters input.

Features plans for the year ahead

This plan tries to outline what could be reasonably and realistically accomplished n the ATF project in the coming year, and beyond, in line with the features and plans outlined in the original project proposal. The feature set in Phase 1 would bring a release quality ATF. (for incubation exit strategy, see below)

Phase 1 Plan:

- unit tests harnesses and test suite
- stabilization, key bug fixing
- add "eval source" debugger support
- move to WTP 3.0
- introduce key experimental APIs and extensions points
- third party library support (JSDT may provide the infrastructure for this, but ATF would be one implementer, or provider)
- packaging and integration with browser (and XULRunner) (pending IP review)

Phase 2 Plan:

- strengthen core
- support for latest Firefox/Xulrunner versions (technical + IP review)
- early cross server-side/client-side JavaScript debugging (with initial targets for JSP and possibly PHP)
- provide WYSIWYG framework editing of UI Components and integrate that with the WTP WYSIWYG editor
- formalize build/test cycle , aligned with WTP

Phase 3 Plan: Incubation Exit requirements

- NLS support, string externalization
- Documentation
- API documentation
- Determine internal/external classes. Rename packages.
- Remove unneeded dependencies, General cleanup of code
- Formalize APIs and Extension points
- Test plans and JUnit test.
- Conform to next release milestone release plan
- Review bug
- Graduation Review/Requirements

Future Features

For the future, the ATF project has ambitious plans to evolve the features set.

Browser Tooling Enhancements

- Browser Tools add save/restore of settings/options
- Allow nodes to be moved inside the DOM Inspector
- Word wrap for request body in XHR monitor
- Option to clear cookies , and other privacy settings in the Browser Editor

Debugger Enhancements

- conditional breakpoint and hit count on breakpoints
- Hover evaluation
- Function entry/exit breakpoints
- Breakpoint line verification
- Debug variables view: options to hide functions, to filter out internal JavaScript properties and to flatten or show 'logical structure' for JS objects
- ScriptView: option to organized the displayed script elements

IE and Webkit support

- Write IE debugger
- Write Webkit embedder
- Write Webkit debugger
- Refactor ATF Code to support multiple browsers
- Update Browser Tools for multi-browser support
- Support DOM Level 3 API for DOM inspector.
- Move Mozilla DOM inspector to DOM Level 3 API.
- Support adapter interfaces for specifying console/monitor
- Support common code for debugger when possible
- Generic Run/Debug launch code.
- Extension point definitions
- Resolve legal issues around IE and Webkit

Server-side support

The goal would be to provide seamless client and server-side debugging when the server side generates and includes JavaScript such that one debug sessions can debug what happens on the server and in the browser. This could require changes in the code of projects such as PDT and/or J2EE Webtools

- Simple debugging from the ATF Debugger to the Server Debugger and would require ATF JavaScript debugger changes and Server-side debugger changes, as well as common launch code
- Support for server-side debuggers such as Xdebug
- Markup generation for lines numbers. This probably requires changes to the server side engine code including its debugger, as well server engine support for generation of markup information to find server-side line numbers from client side, creattion of client side/Server side markers, and ATF JavaScript debugger changes and Server-side debugger changes

JavaScript Code Profiler

- Provide/borrow UI for profiling, tie to Mozilla's jsdIScript profiling hooks
- Add I.E. and Webkit support

Community and communication plans

On the one hand, there has been a lot of interest to build and expand on the browser-based tools provided by ATF. On the other hand, Ajax and JavaScript are becoming mainline technologies and still lack badly tooling support. ATF is uniquely positioned to provide a strong Eclipse-based support for Ajax and JavaScript tooling.
The plan is to continue the team efforts in terms of community and communication and increase those further with blogs, more predictable builds, and more accessible tools. As part of the communication related changes, we may consider a project name changes such that the project name is more self-explanatory. Some early community polling were conducting on that topic last year and could be used as a base.

Incubation exit plans

The plan for exiting incubation is be to be integrated in the Web tools platform. Since the Ganymede release is already underway, the plan would be to start integration after Ganymede has been released, toward possible early integrated WTP drops around Q1 2009. The features and task outlined in Phase 1 and 2 of the feature plan would bring ATF to release quality, to integrate ATF as a WTP component.

Known Adopters and Interested Parties

- Aptana: builds on ATF code
- Codegear: incorporates and builds on ATF for JavaScript and Ajax tooling
- Genuitec: builds on ATF code
- Nexaweb: incorporates and builds on ATF for JavaScript and Ajax tooling
- nexB: incorporates and builds on ATF for JavaScript and Ajax tooling
- Redhat/JBoss : builds on ATF for JSF and HTML tooling
- Suprasphere: builds on ATF code for financial research tools

New committers coming aboard with these new plans and leadership

Giuliano Mega (nexB)
Giuliano, currently with nexB, has been one of the few selected students for the Eclipse Summer of Code 2006. He delivered there a sophisticated distributed debugger based on Eclipse. Giuliano has been starting contributing to ATF with the creation of a multi-threaded JUnit test harness for embedded browser-based feature testing and early work on the eval support in the debugger.

Laurens VandePut (independent and http://joomlatools.org/)
Laurens has been one of the few selected students for the Joomla Google Summer of Code 2006 and was a mentor there in 2007. He wrote an Eclipse browser-based WYSIWYG edit, and wants to contribute this browser-based tool to the ATF project to evolve it as framework for WYWISWYG Ajax editors building.

Reaching out: beyond those two potential new committers, there are opportunities to reach out to welcome and integrate the efforts of some related open source Eclipse project in the domain of debugging (such as Firebug) and projects that provide ATF-enabled browser based tools and frameworks, as well as commercial adopters.