Runtime Infrastructure - Feature #2594

Implement a tracing facility for P2J

06/24/2015 11:04 AM - Hynek Cihlar

Status:	New	**Start date:**	06/24/2015
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

History

#1 - 06/24/2015 12:08 PM - Hynek Cihlar

When trying to understand internal workings of a complex system like P2J - spanning multiple hosts, processes and threads - it may come handy to trace the execution path and gather related runtime information like method argument and return values.

The idea is to create a tracing facility that would be active during runtime of the target JVM process and would allow the developer to trace specific parts of the target application - a facility like "DTrace for P2J".

Here is a set of features and requirements the implemented facility should conform to:

- Data driven approach - the facility will be fed with expressions that will allow the developer to "cherry pick" the desired set of methods/classes/packages to trace.
- If possible, the facility should be capable to trace reads/writes of class fields. This would be useful, for example, for the P2J client-server synchronization subsystem (config classes).
- The facility will allow to start or pause tracing during runtime. The amount of trace output may be substantial, also it may be useful to give the tracing output a scope.
- The tracing output entries will include method argument values and return values.
- The facility will format common P2J data types to human-readable strings.
- The tracing output entries will include timestamps in nanosecond resolution.
- A nice to have feature would be tracing profiles - sets of tracing expressions that would be somehow logically related. Tracing profiles would be managed centrally for all developers and so the individual developer would only choose the desired tracing profile - "Events", "Drawing", "Client/server communication", "Client/server state synchronization" - instead of having to manually create the tracing expressions himself.
- The tracing output will go to the default P2J logging facility and optionally to other outputs and formats - CSV files for example.
- The tracing facility should not incur any overhead when not configured (or disabled) but still would be nice to allow tracing in the post-devel environments.

In terms of implementation, an AspectJ looks like the simplest way of "point cutting" the running application, but it is slow and limits to projects with AspectJ support. Java byte code instrumentation seems to be a better way to go. We already use ASM and BCEL. Also consider to use BTrace which already implements great deal of the core tracing features.

See #2534 (notes 141-143, 145, 148, 151-153) where the idea originated.

#2 - 06/24/2015 12:20 PM - Greg Shah

- Project changed from FWD to Runtime Infrastructure

#3 - 11/16/2016 01:08 PM - Greg Shah

- Target version deleted (Deployment and Management Improvements)