Runtime Infrastructure - Feature #1848

make logging settings controllable dynamically at runtime

10/30/2012 02:14 PM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	80.00 hours
Target version:	Deployment and Management Improvements		
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to Runtime Infrastructure - Bug #5703: rationalize, standardize and s...	**Closed**

History

#1 - 10/31/2012 11:49 AM - Greg Shah

- Target version set to Deployment and Management Improvements

#2 - 11/16/2016 01:06 PM - Greg Shah

- Target version changed from Deployment and Management Improvements to Deployment and Management Improvements

#3 - 06/21/2019 09:17 AM - Greg Shah

The core objective is to allow an administrator to change the detailed logging levels at runtime, through the admin console.

As part of this work, these are some issues to address:

- All logging must be done with the standard FWD logging helpers and configuration. Today there is some usage of log4j (mostly in persistence) and Apache commons (search on imports of org.apache.commons.logging).
- Most places that log calculate the logging levels once (when the class loads). I think the easiest/cleanest approach:
  - Create a helper class which can be instantiated to hide the management and state of the logger.
  - Make this helper a listener where it registers for a callback when the logging level changes.
  - Expose simple APIs for common usage.
- Create an admin UI for displaying/editing the logging levels and logging settings as they can be set in the directory.
  - Changes must be saved in the directory.
  - Fire the listener to change the state of existing loggers.

#4 - 06/21/2019 10:14 AM - Igor Skornyakov

log4j exposes its loggers' settings via JMX, so there is no need for custom instrumentation. If we use log4j commons logging bridge than it will work for JCL as well.

#5 - 04/27/2023 05:06 AM - Galya B

- Related to Bug #5703: rationalize, standardize and simplify the client-side log file name configuration added

#6 - 04/27/2023 05:09 AM - Galya B

A note: For dynamic levels to apply to all child packages and classes, all loggers should have fully qualified names (not simple names, check usage of class.getSimpleName()). Root logger (named "") level applies to all loggers, even those with simple names (when these are not set explicitly).

#7 - 04/27/2023 05:23 AM - Galya B

A note: With #5703 all levels can be changed dynamically. To not lose logs, when the level is lowered, isLoggable() should not be cached on the class or instance levels. Only admin menu is to be implemented.

#8 - 04/27/2023 08:26 AM - Greg Shah

> A note: With #5703 all levels can be changed dynamically. To not lose logs, when the level is lowered, isLoggable() should not be cached on the class or instance levels. Only admin menu is to be implemented.

For performance reasons we often cache the result of isLoggable(). This should be maintained. We can easily do this by passing an optional listener to the LogHelper.getLogger() call which will be notified if the logging level changes for that logger.

#9 - 05/10/2023 08:04 AM - Galya B

Greg Shah wrote:

> For performance reasons we often cache the result of isLoggable(). This should be maintained. We can easily do this by passing an optional listener to the LogHelper.getLogger() call which will be notified if the logging level changes for that logger.

isLoggable results are now cached in CentralLogger and invalidated when parent levels change. I tried to remove all class / object level cached results.

#10 - 05/10/2023 08:06 AM - Galya B

Client levels invalidation needs more work, check #5703-356 point 2.