What's new in Quartz 2.0

What's New In Quartz Scheduler 2.0

(As Of Beta 1, November 2010)

This is the most significant revision of Quartz Enterprise Job Scheduler since the project began over decade ago. Efforts have been made to strike a balance between preserving backward compatibility (or at least providing a fairly easy migration) and providing meaningful long-wanted changes to the API.
API Changes
The most obvious differences with version 2.0 are the significant changes to the API. These changes have aimed to: modernize the API to use collections and generics, remove ambiguities and redundancies, hide/remove methods that should not be public to client code, improve separation of concerns, and introduce a Domain Specific Language (DSL) for working with the core entities (jobs and triggers).

While the API changes are significant, and the usage of the "new way of doing things" is highly encouraged, there are some formulaic (search-and-replace) techniques that can be used to get 1.x code quickly working with version 2.0. See the migration guide for more information.

Outline of most significant API changes:

- API methods that return (or take as parameters) arrays now return (or take) typed collections. For example, rather than getJobGroupNames(): String[] we now have getJobGroupNames(): List<String>
- Job and Trigger identification is now based on JobKey and TriggerKey. Keys include both a name and group. Methods which operate on particular jobs/triggers now take keys as the parameter. For example, getTrigger(TriggerKey key): Trigger, rather than getTrigger(String name, String group): Trigger.
- New DSL/builder-based API for construction Jobs and Triggers:

```
JobDetail job = newJob(SimpleJob.class)
.withIdentity("job1", "group1")
.build();
Trigger trigger = newTrigger()
.withIdentity("trigger1", "group1")
.startAt(futureDate(2, IntervalUnit.HOURS))
.withSchedule(repeatHourlyForever())
.modifiedByCalendar("holidays")
.build();
```

- Methods from TriggerUtils related to easy construction of Dates have been moved to new DateBuilder class, that can be used with static imports to nicely create Date instances for trigger start and end times, etc.

```
// build a date for 10:00 am on Halloween
Date  runDate = dateOf(0, 0, 10, 31, 10);
// build a date 2 hours in the future
Date myDate = futureDate(2, IntervalUnit.HOURS);
```

- The StatefulJob interface has been deprecated in favor of new class-level annotations for Job classes (using both annotations produces equivalent to that of the old StatefulJob interface):
  - PersistJobDataAfterExecution - instructs the scheduler to re-store the Job's JobDataMap contents after execution completes.
  - DisallowConcurrentExecution - instructs the scheduler to block other instances of the same job (by JobKey) from executing when one already is.
- New annotation: ExecuteInJTATransaction. Adding this annotation to a Job class instructs Quartz to start a JTA transaction before executing the job (and commit/rollback after completion/exception). The configuration property 'wrapJobExecutionInUserTransaction' from version 1.x still exists, but the new annotation lets you tune the behavior on per job, while the config property affects all jobs.
- Significant changes to usage of JobListener and TriggerListener:
  - Removal of distinction between "global" and "non-global" listeners
  - JobDetails and Triggers are no longer configured with a list of names of listeners to notify, instead ** ** listeners identify which jobs /triggers they're interested in.
  - Listeners are now assigned a set of Matcher instances - which provide matching rules for jobs/triggers they wish to receive events for.
  - Listeners are now managed through a ListenerManager API, rather than directly with the Scheduler API.
- The SchedulerException class and class hierarchy has been cleaned up.
- DateIntervalTrigger was renamed to CalendarIntervalTrigger (or more exactly the concrete class is now CalendarIntervalTriggerImpl).
- The notion of "volatility" of jobs and triggers has been eliminated.
- New trigger misfire instruction MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY lets a trigger be configured in such a way that it is selectively ignored from all misfire handling. In other words, it will fire as soon as it can, with no special handling - a great option for improving performance particularly with setups that have lots of one-shot (non-repeating) triggers.

New Features

- Scheduler.clear() method provides convenient (and dangerous!) way to remove all jobs, triggers and calendars from the scheduler.
- Scheduler.scheduleJobs(Map<JobDetail, List<Trigger>> triggersAndJobs, boolean replace) method provides convenient bulk addition of jobs and triggers.
- Scheduler.unscheduleJobs(List<TriggerKey> triggerKeys method provides convenient bulk unscheduling of jobs.
Scheduler.deleteJobs(List<JobKey> jobKeys)method provides convenient bulk deletion of jobs (and related triggers).

- Scheduler.checkExists(JobKey jobKey)and Scheduler.heckExists(TriggerKey triggerKey)methods provides convenient way to determine uniqueness of job/trigger keys (as opposed to old have of having to retrieve the job/trigger by name and then check whether the result was null).
- JDBCJobStore now allows one set of tables to be used by multiple distinct scheduler instances
- JDBCJobStore is now capable of storing non-core Trigger implementations without using BLOB columns, through the use of the new TriggerPersistenceDelegate interface, which can (optionally) be implemented by implementers of custom Trigger types.
- JDBCJobStore now includes a SybaseDelegate for improved compatibility with Sybase
- Cron expressions now support the ability to specify an offset for "last day of month" and "last weekday of month" expressions. For examples: "L-3" (three days back from the last of the month) or "L-3W" (nearest weekday to the day three days back from the last day of the month).
- XML files containing scheduling data now have a way to specify trigger start times as offsets into the future from the time the file is processed (useful for triggers that need to begin firing some time after the application is launched/deployed).
  From schema: <xs:element name="start-time-seconds-in-future" type="xs:nonNegativeInteger"/>
- XML file schema now supports specifying the 'priority' property of triggers.
- QuartzInitializerListener (and QuartzInitializerServlet) supports new parameter "wait-on-shutdown", which will cause the Scheduler to wait for executing jobs to complete before shutting down, when the Servlet container is un-deploying the application.

Miscellaneous

- Various performance improvements, including (but not limited to):
    - Ability to batch-acquire triggers that are ready to be fired, which can provide performance improvements for very busy schedulers
    - Methods for batch addition/removal of jobs and triggers (see "New Features")
- Various bug fixes, for complete listing see the release notes from Jira: https://jira.terracotta.org/jira/secure/ReleaseNote.jspa?projectId=10282&version=10842
- All examples packaged in the "examples" directory of the Quartz distribution have been updated to make use of (demonstrate) the new API for defining jobs and triggers.