CoP: Scheduler Options

An introduction to the Quartz framework

Our Functional Requirements

- ensure custom domains always have a valid SSL certificate
  - regularly check for approaching expiry
  - renew expiring certificates in time

Our Non-Functional Requirements

- rate limit: spread renewal jobs over 24h
- don't trigger renewal jobs on each cluster node
- track renewal job state for reliability

Quartz Job Scheduler

- very popular Open Source Java library
- Custom Spring Boot Starter available

Setup

- create a set of DB tables
- add dependency to `de.chandre.quartz:spring-boot-starter-quartz`
- customize SchedulingConfiguration
  - enable clustered mode
  - configure DB access
  - configure thread pool (size depends on DB conn pool)
  - create `JobDetail` for expiry check
  - create `Trigger` based on cron expression

Implementing `Job` interface

- extend `QuartzJobBean` to allow dependency injection
  - also for runtime properties via `JobDataMap` state
- implement
  `void executeInternal(JobExecutionContext)`
  - access input data via
    `JobDataMap getMergedJobDataMap()`
  - store output data via `void setResult(Object)`
  - rethrow as `JobExecutionException` to control unscheduling or re-firing of failed job

Programmatic scheduling

- prepare `JobDetail` via `JobBuilder`
  - `Job` class for job execution
  - durability and recovery
  - key/value pairs in `JobDataMap`
- prepare `Trigger` via `TriggerBuilder`
  - choose `ScheduleBuilder`
    - simple, cron, calendar interval, daily time interval
  - start/end date
  - key/value pairs in `JobDataMap`
- schedule execution using `Scheduler#scheduleJob(JobDetail, Trigger)`