Job Plugins

See also:

- Writing Registry Plugins
- Iterating Large Datasets
- Registry Jobs

Some additional conventions are required when writing a Job Plugin.

1. The name of the Plugin should match the format `FooJob`.
    a. For Registry v3.3.x, the prefix (eg: `Foo`) will be used as the `job_type` in cm_co_jobs.
    b. ℹ As of Registry v4.0.0, Job Plugins are no longer required to follow the `FooJob` naming convention for the Plugin itself. This enables polymorphic Plugins. ie: Other types of Plugins can also implement Jobs.
2. For Registry v3.3.x, the Plugin must implement a single model `FooJob` (in `Model/FooJob.php`), where *Foo* matches the name of the Plugin.
3. For Registry v4.0.0 and later, the Plugin may implement multiple Job models, each of which follows the naming convention `BarJob`, where *Bar* may be any string.
    a. The main Plugin model (the model whose name matches the name of the Plugin) must implement an additional function:
        i. getAvailableJobs(), which returns an array whose keys are the prefixes for each defined Job, and whose values are help strings describing the Job. Prefixes should be in CamelCase (eg: CleanTable).
    b. The `job_type` in cm_co_jobs is the desired plugin Job model in Cake notation, but without the *Job* suffix, eg: `FooJob.Bar`.
4. Each Job Model should extend `CoJobBackend`, which defines some standard interfaces and provides some behind the scenes common functionality.
    a. Each Model must implement two functions:
        i. `execute($coId, $CoJob, $params)`, which will be passed the relevant CO ID, a CoJob object, and an (optional) array of parameters in accordance with the Plugin's configuration.
            1. When the Job is started, it should call `$CoJob->update()` (with appropriate arguments).
            2. When executing, the Job should not typically generate output, but should instead create Job History records. If it is absolutely necessary to write to stdout, the Job should follow Cake's Console output levels.
            3. If the Job loops over multiple records, it should check `$CoJob->canceled($jobid)` after each iteration, and immediately terminate processing if the Job has been canceled.
            4. Jobs may provide a "percent complete" value so that the job status view page will generate a progress bar. Simply call `$CoJob->setPercentComplete($jobid, $percent)` at appropriate points during processing.
            5. When complete, the Job must call `$CoJob->finish()`, with appropriate arguments. (The Job will be set to started when dispatched, before the Plugin is invoked.)
        ii. `parameterFormat()`, which returns an array of parameters supported by the plugin. Each entry in the array has a key of the parameter name and a value of an array with the following keys:
            1. `choices`: An array of permitted values for the parameter (when `type` is `select`)
            2. `help`: Help text for the parameter
            3. `required`: Boolean indicating whether or not this parameter is required
            4. `type`: One of `bool`, `int`, `select`, or `string`.
    b. As of Registry v4.0.0, `CoJobBackend.php` sets `$useTable = false`. Jobs that use a table may need to override this.

⚠ Job Plugins, including any code they call, should be careful not to trigger session creation, since sessions in a shell or cron context do not necessarily make sense. If code is mixed use (ie: may also be called from the UI), be sure to check that the session has been created before trying to read from it.

```
if(session_status() == PHP_SESSION_ACTIVE) {
  $foo = CakeSession::read('Foo.foo');
} else {
  // No session, in a shell command
  $foo = DEFAULT_VALUE;
}
```