Invoke Rake Task inside Sidekiq Cron Worker

Tags	
Published	☑

Running schedule process using sidekiq cron was a breeze. The setup was easy, you just need to add sidekiq-cron gem inside your gem file and add schedule.yml to define the cron schedule definition. And also it includes the a page to view list of registered cron within the sidekiq dashboard.

Usually, when running a schedule process we create a sidekiq worker for each cron and inside the workers we invoke another service or rake task.

```
daily_backup_worker:
  cron: "0 0 * * *"
  class: "DailyBackupWorker"
  description: "Run daily backup at midnight"
weekly_backup_worker:
    cron: "0 0 * * 0"
    class: "WeeklyBackupWorker"
    description: "Run weekly backup on Sunday at midnight"
```

Inside the worker class for example `DailyBackupWorker` we call the existing rake task command `db:backup` to trigger the backup database.

```
require 'rake'
Rails.application.load_tasks

class DailyBackupWorker
    include Sidekiq::Worker

    def perform
        Rake::Task['db:backup'].execute
```

```
    end
end
```

Moreover, the `WeeklyBackupWorker` class probably will have similar code with `DailyBackupWorker`. What if I told you that we can use a single class to handle rake task invocation inside sidekiq cron worker?

Here the example, instead creating each class to handle each sidekiq-cron schedule. we can create a class called `InvokeRakeTaskWorker`

```
require 'rake'
Rails.application.load_tasks

class InvokeRakeTaskWorker
    include Sidekiq::Worker

    def perform(command)
        Rake::Task[command['task']].execute(command['args'])
    end
end
```

And you can change your schedule.yml into like this.

```
daily_backup_worker:
  cron: "0 0 * * *"
    class: "InvokeRakeTaskWorker"
    args:
        task: "db:backup"
  description: "Run daily backup at midnight"
weekly_backup_worker:
    cron: "0 0 * * 0"
    class: "InvokeRakeTaskWorker"
    args:
        task: "db:backup"
        args: "--weekly"
    description: "Run weekly backup on Sunday at midnight"
```

Conclusion

That it’s, now you can use single class to invoke rake command inside sidekiq cron schedule.

Using a single worker class to handle the invocation of rake tasks with different arguments brings code reusability, simplified configuration, flexibility, easier maintenance, and improved testing capabilities to your Rails application's background job scheduling.

It's important to carefully evaluate the trade-offs and consider the specific requirements and complexity of your application before deciding whether to use a single worker class or separate worker classes for your cron jobs.