

# Jira Java에서 Scheduler 구현하기

이 문서는 Java에서 Jira Scheduler를 구현하는 방법을 공유하기 위해 작성되었다.

도구명	jira, java
버전	8.22

- [Scheduler register, unregister](#)
  - [onStart, onStop Override](#)
  - [scheduleService 응용](#)
  - [JobId, JobRunnerKey 등록](#)
- [Jira Scheduler 동작 확인](#)
  - [Jira Scheduler menu](#)
- [참조](#)

## Scheduler register, unregister

### onStart, onStop Override

```
@ExportAsService
@Named
public class ScheduleSync implements LifecycleAware {

    @ComponentImport
    private final SchedulerService schedulerService;

    private final JobId ISSUE_SYNC_JOB_ID = JobId.of("issue-sync-job-id");

    @Override
    public void onStart() {
        issueSyncService.doLogin();
        JobConfig jobConfig = JobConfig.forJobRunnerKey(ScheduleJob.ISSUE_SYNC_JOB_RUNNER_KEY)
            .withSchedule(Schedule.forCronExpression((String) getDbInfo().get("cron")))
            .withRunMode(RunMode.RUN_LOCALLY);
        schedulerService.scheduleJob(ISSUE_SYNC_JOB_ID, jobConfig);
    }

    @Override
    public void onStop() {
        schedulerService.unscheduleJob(ISSUE_SYNC_JOB_ID);
    }
}
```

- LifecycleAware의 plugin onStart, onStop 메소드 구현
- onStart : plugin이 시작될 때 scheduleService api - Jira scheduler register (do Job)
- onStop : plugin이 종료되면 scheduleService api - Jira scheduler unregister (stop Job)

## scheduleService 응용

```

public void controlJob() throws SchedulerServiceException {
    if(getDbInfo().get("scheduleJob").equals("on")){
        doJob();
    }else if(getDbInfo().get("scheduleJob").equals("off") || getDbInfo().get("scheduleJob") == null){
        stopJob();
    }
}

```

- 서버 DB에 Scheduler정보를 저장, 임의적으로 register, unregister handling
- plugin이 작동되는 동안 scheduler on/off 가능 (servlet - templateRenderer로 vm, js 형식으로 UI 기능 구현)

## JobId, JobRunnerKey 등록

```

public class ScheduleJob implements JobRunner {
    public static final JobRunnerKey ISSUE_SYNC_JOBRUNNER_KEY = JobRunnerKey.of(ScheduleJob.class.
getCanonicalName());
}

public class ScheduleSync implements LifecycleAware {
    private final JobId ISSUE_SYNC_JOB_ID = JobId.of("issue-sync-job-id");

    JobConfig jobConfig = JobConfig.forJobRunnerKey(ScheduleJob.ISSUE_SYNC_JOBRUNNER_KEY)
        .withSchedule(Schedule.forCronExpression((String) getDbInfo().get("cron")))
        .withRunMode(RunMode.RUN_LOCALLY);
    schedulerService.scheduleJob(ISSUE_SYNC_JOB_ID, jobConfig);
}

```

- ScheduleJob, ScheduleSync class에 각 저장한 JobRunnerKey, JobId를 파라미터로 Schedule Job Run

## Jira Scheduler 동작 확인

### Jira Scheduler menu

The screenshot shows the Jira Administration interface. The 'Database Connection' section is visible, with fields for JDBC URL, User, and Password. Below this, the 'Schedule INFO' section is highlighted with a red box. It contains a CRON expression field with the value '0/1 \* \* \* \* ?' and a 'Schedule on/off' toggle switch that is currently turned on. There are also 'Save' and 'Delete' buttons at the bottom of the 'Schedule INFO' section.

- DB에 CRON 표현식과 Scheduler register 여부 저장

The screenshot shows the Jira Scheduler Admin interface. On the left is a sidebar with navigation links. The main area displays a table of scheduled jobs. One job, 'com.curvc.atlas.jira.plugin.schedule.ScheduleJob', is highlighted with a red box. Below the job name, its details are shown:

#1	issue-sync-job-id	실행 가능
	유형:	0
	파라미터:	로컬에서
	실행 모드:	16 * * * * ?
	다음 실행:	Mon Aug 08 13:31:16 KST 2022
	다음 실행 시간:	62 밀리초
	다음 실행:	Mon Aug 08 13:29:12 KST 2022
	메시지:	

Below the details, there is a table of jobs with columns for job name, frequency, and a '더 보기' (More) link. The highlighted job is 'com.curvc.atlas.jira.plugin.schedule.ScheduleJob' with a frequency of '1 작업' and 'CRON'.

- 시스템 - 스케줄러 상세정보 - 설정한 scheduler key 검색
- 추가한 Scheduler의 Key와 다음 실행될 일정에 대한 정보

## 참조

- [Jira SchedulerService api](#)