|  |
| --- |
|  |
| chamomile-batch Architecture |
| 배치실행환경 아키텍처 |
|  |
|  |
|  |

|  |
| --- |
|  |

목차

[1. 배치실행 어플리케이션 구성 2](#_Toc512034029)

[2. 배치실행 어플리케이션 디렉터리 구조 2](#_Toc512034030)

[3. 배치실행 어플리케이션 Startup and Shutdown 3](#_Toc512034031)

[4. 배치프로그램 개발 3](#_Toc512034032)

[5. 배치프로그램 테스트(JUnit) 3](#_Toc512034033)

[6. 배치프로그램 배포 4](#_Toc512034034)

[7. 배치실행환경 Workflow등록 4](#_Toc512034035)

[8. 배치프로그램 실행 5](#_Toc512034036)

[ **Scheduler에 등록하여 실행** 5](#_Toc512034037)

[ **RESTful API 호출로 실행** 5](#_Toc512034038)

[ **배치실행 어플리케이션 관리자UI를 이용한 실행** 5](#_Toc512034039)

[ **CLI (Command Line Interface)를 이용한 실행** 5](#_Toc512034040)

[9. 배치프로그램 로그파일 조회 6](#_Toc512034041)

[10. 배치프로그램 실행이력 조회 및 재실행 6](#_Toc512034042)

[11. 배치실행 어플리케이션 데이터 모델 6](#_Toc512034043)

# 배치실행 어플리케이션 구성

배치실행 어플리케이션이란 개발된 배치프로그램의 스케줄링 및 실행을 하기 위한 Java Standalone 어플리케이션으로(웹과 비교하자면 배치실행 어플리케이션은 웹서버, 각 배치프로그램은 각각의 웹화면에 비교할수 있음) 웹 기반 배치프로그램 실행 및 관리를 위한 Tomcat 웹서버 및 배치프로그램에서의 데이터베이스 연동을 위한 Mybatis가 Embedded된 형태의 독립 어플리케이션이다. 대략적인 내부 구조는 다음과 같다.



# 배치실행 어플리케이션 디렉터리 구조

Deploy된 디렉터리 및 파일 구조는 아래와 같다.

|  |  |
| --- | --- |
| **디렉토리** | **설명** |
| ./conf | 설정파일 디렉토리 |
| ./conf/application.xml | 어플리케이션 설정파일 |
| ./conf/application.properties | 설정파일(port,데이터베이스접속정보 등) |
| ./conf/mybatis.xml | MYBATIS 설정파일 |
| ./webapp/application/\* | Embedded Tomcat에서 제공되는 웹어플리케이션(관리화면,RESTful API) |
| ./application.sh | 어플리케이션 구동 쉘스크립트 |
| ./chamomile-batch-${version}.jar | 어플리케이션 JAR 바이너리 파일(배치프로그램 개발시에도 해당 libray 사용) |
| ./build.sh | 빌드 자동화 스크립트(개발환경에서만 사용) |
| ./run-batch.sh | 배치Workflow를 Command로 실행하기 위한 쉘스크립트 |
| **./depoly** | **개발된 배치프로그램의 jar가 deploy되는 디렉토리** |

# 배치실행 어플리케이션 Startup and Shutdown

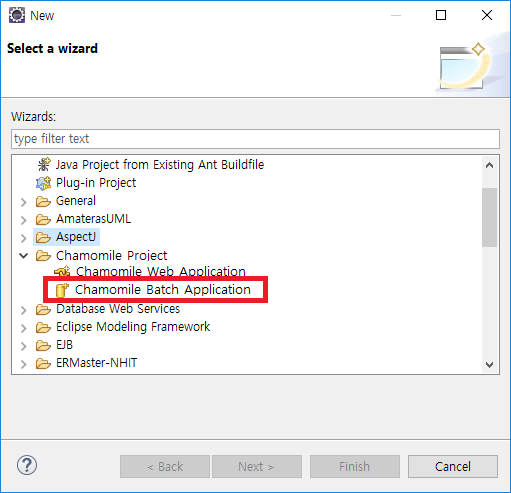
# 배치실행 어플리케이션 구동  
shell> ./application.sh start

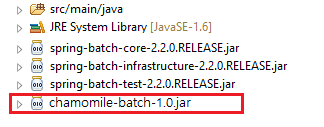
# 배치실행 어플리케이션 프로세스 조회  
shell> ./application.sh status

# 배치실행 어플리케이션 종료  
shell> ./application.sh stop

# 배치프로그램 개발

배치프로그램의 개발인 기본적으로 **spring-batch-${version}.jar** 및 **chamomile-batch-${version}.jar** 라이브러리로 이클립스IDE 환경에서 개발한다. (추후 이클립스 프로젝트 템플릿 플러그인이 개발되면 신규프로젝트 생성 템플릿에서 chamomile-batch 프로젝트를 선택하면 기본적으로 spring-batch와 chamomile-batch Dependency 추가된 프로젝트 템플릿이 생성)



  
***PS. 상세내용은 [배치프로그램 개발가이드]를 참조.***

# 배치프로그램 테스트(JUnit)

각 프로젝트에서 개발된 spring-batch 프로그램은 **chamomile-batch-${version}.jar**에서 제공되는 **ApplicationJUnit4ClassRunner**를 사용하여 아래와 같이 JUnit테스트를 수행한다.

@RunWith(ApplicationJUnit4ClassRunner.**class**)

@ContextConfiguration(locations="file:src/test/{project}/spring-test.xml")  
**public** **class** BatchTest {

@Test

Public void testBatch {

JobCallable jobCallable = JobCallableFactory.newJobCallable(SPRING);

jobCallable.setCommand(“[spring-batch.xml]$[job-id]”);

jobCallable.setInput(“입력값”);

jobCallable.exeucte(); // 배치프로그램 실행

}

}

***PS. 상세내용은 [배치프로그램 개발가이드]를 참조.***

# 배치프로그램 배포

프로젝트에서 개발된 배치프로그램(XXX.jar)은 업무단위 별 JAR 또는 프로젝트배치 전체 JAR를 FTP로 ${배치실행어플리케이션설치디렉터리}/deploy 디렉터리에 업로드 한다. (배치실행 어플리케이션에서 동적으로 로딩)

**${배치실행어플리케이션설치디렉터리}/deploy** 🡸 디렉터리에 복사

# 배치실행환경 Workflow등록

개발된 배치프로그램을 배치실행 어플리케이션을 통해 실행하기 위하여 관리자화면에서 Workflow의 Job으로 등록한다.



WORKFLOW 입력항목

|  |  |
| --- | --- |
| ID | Workflow식별아이디 |
| Name | Workflow의 업무명 |
| Cron Expression | 실행할 스케줄 Cron Expression(추후 개발 시 가독성 있는 입력형식으로 변경) |
| Parameter | 해당 Workflow에서 사용할 Parameter를 정의한다.(영업일자 등) | |
| Description | Workflow의 설명을 기술한다. | |
| Use or Not | 사용여부 설정 | |
|  |  | |

JOB 입력항목

|  |  |
| --- | --- |
| ID | 배치Job의 식별아이디 |
| Name | 배치Job 이름 |
| Type | **SPRING** : spring-batch 기반으로 구현된 배치프로그램  **JAVA**: JAVA 어플리케이션(spring-batch가 아닌 chamomile JavaJobCallable 인터페이스를 상속하여 개발된Java 프로그램)  **CLI**: Command Line Interface (파일이동,로그삭제등의 쉘명령어 또는 pro\*c등 독자적으로 개발된 배치프로그램 실행 시) |
| Command | 각 Type별 실행할 프로그램의 Command를 입력  **Type이 SPRING인 경우 : [spring-batch.xml]$[job id]**  예> com/company/project/batch/lms/LMS001.xml$extractCustomerJob  **Type이 JAVA인 경우: [class path]** 예> com.company.project.batch.lms.LMS001.java  **Type이 CLI인 경우: [shell command]**  예>/home/company/project/batch/LMS001.sh | |
| Argument | 프로그램 실행 시 입력될 패라미터를 key,value 형태로 정의 | |
|  |  | |

# 배치프로그램 실행

배포된 배치프로그램을 Workflow 등록 후 실행하는 방법은 아래와 같다.

## **Scheduler에 등록하여 실행**

배포된 배치프로그램을 관리자UI에서 스케줄링 하여 실행

## **RESTful API 호출로 실행**

HTTP Request를 이용하여 실행(통합 관리자 화면 및 타 시스템 연동 시)

예>http://${배치실행서버}/api/workflow/${Workflow아이디}/execute 호출

## **배치실행 어플리케이션 관리자UI를 이용한 실행**

http://${배치실행서버}/admin 으로 로그인 후 [WorkflowManage] 메뉴로 이동 후 [RUN]버튼 실행

## **CLI (Command Line Interface)를 이용한 실행**

배치실행 어플리케이션 서버 SSH접속 후 해당 command로 실행

host> run-batch.sh [Workflow아이디] [JOB아이디] [input parameter]

# 배치프로그램 로그파일 조회

배치프로그램의 로그는 오류 시 각 배치프로그램 별로 확인이 가능하여야 하므로 배치실행 어플리케이션의 로그와는 별도로 각 실행 단위 별로 출력한다. 각 배치프로그램의 로그파일 위치 및 Naming룰은 아래와 같다.

shell> tail -F ${APP\_HOME}/log/**[실행\_일련번호]\_ [Workflow아이디].log**

# 배치프로그램 실행이력 조회 및 재실행

배치프로그램의 실행이력은 배치실행 어플리케이션의 관리자 화면에서 확인 및 재실행이 가능하다.

**PS. 추후 화면 개발**

# 배치실행 어플리케이션 데이터 모델

**PS. 배치실행 어플리케이션 ERD 참조**