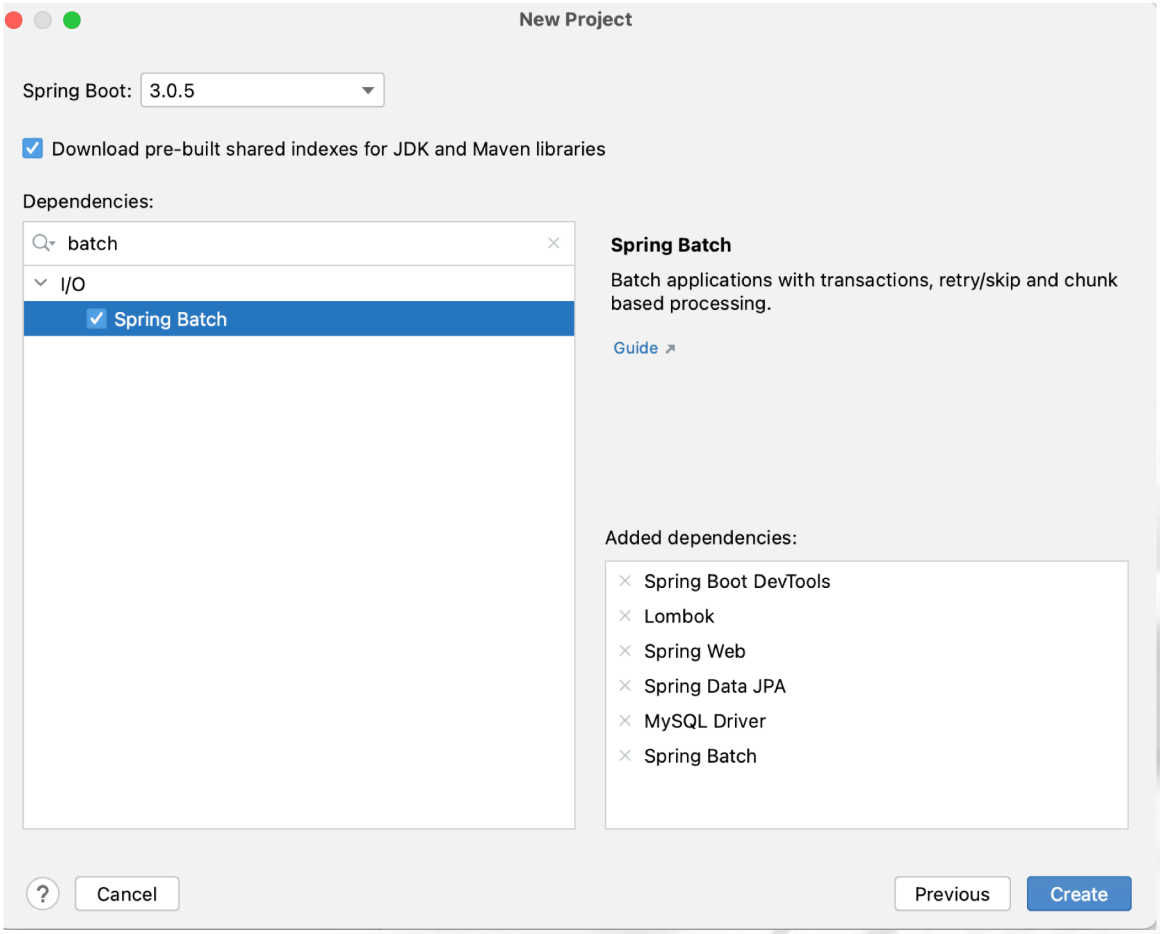
Spring Batch

Spring Batch

❖ Open API 데이터를 읽어서 데이터베이스에 저장하는 배치 프로젝트

✓ 프로젝트 생성



Spring Batch

- Open API 데이터를 읽어서 데이터베이스에 저장하는 배치 프로젝트
  - ✓ build.gradle 에 의존성 추가

```
implementation 'org.json:json:20230227'
```

Spring Batch

- Open API 데이터를 읽어서 데이터베이스에 저장하는 배치 프로젝트
  - ✓ application.yml 파일에 데이터베이스 경로 설정

```
#? ?? ?? ?? ??
server:
  port: 80

#?????? ?? ??
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/adam?serverTimezone=Asia/Seoul
    username: root
    password: wnddkd
  jpa:
    hibernate:
      ddl-auto: update
  batch:
    jdbc:
      initialize-schema: always
```

Spring Batch

- Open API 데이터를 읽어서 데이터베이스에 저장하는 배치 프로젝트
  - ✓ Entity 클래스 생성

```
@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Table(name = "book")
public class Book {
    @jakarta.persistence.Id
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String title;

    private int price;

}
```

Spring Batch

❖ Open API 데이터를 읽어서 데이터베이스에 저장하는 배치 프로젝트

✓ Batch 클래스 생성

```
@Configuration
public class BatchConfiguration {

    private final String key = "88d96a5898edf41ad15f0282f002e213";
    private List<Book> collectData = new ArrayList<>(); //Rest로 가져온 데이터를
리스트에 넣는다.
    private int nextIndex = 0;//리스트의 데이터를 하나씩 인덱스를 통해 가져온다.

    public BatchConfiguration(){
        try {

            String urlAddr =
"https://dapi.kakao.com/v3/search/book?target=title&query=";
            //파라미터 인코딩
            String title = "삼국지";
            urlAddr += URLEncoder.encode(title, "utf-8");
            URL url = new URL(urlAddr);
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            //요청 헤더 설정
            conn.setRequestProperty("Authorization", "KakaoAK
88d96a5898edf41ad15f0282f002e213");
```

Spring Batch

- Open API 데이터를 읽어서 데이터베이스에 저장하는 배치 프로젝트
  - ✓ Batch 클래스 생성

```
StringBuilder sb = new StringBuilder();
if (conn != null) {
    conn.setConnectTimeout(20000);
    conn.setUseCaches(false);
    if (conn.getResponseCode() == HttpURLConnection.HTTP_OK) {
        InputStreamReader isr = new
InputStreamReader(conn.getInputStream());
        BufferedReader br = new BufferedReader(isr);
        while (true) {
            String line = br.readLine();
            if (line == null) {
                break;
            }
            sb.append(line);
        }
        br.close();
        conn.disconnect();
    }
}
String str = sb.toString();
System.out.println(str);
```

Spring Batch

❖ Open API 데이터를 읽어서 데이터베이스에 저장하는 배치 프로젝트

✓ Batch 클래스 생성

```
                JSONObject obj = new JSONObject(str);

                JSONArray documents = obj.getJSONArray("documents");
                for (int i = 0; i < documents.length(); i++) {
                    JSONObject document = documents.getJSONObject(i);

collectData.add(Book.builder().title(document.getString("title")).price(document.getInt
("price")).build());
                }
            }catch(Exception e){}
        }
```

Spring Batch

- Open API 데이터를 읽어서 데이터베이스에 저장하는 배치 프로젝트
  - ✓ Batch 클래스 생성

```
//Rest API로 데이터를 가져온다.
@Bean
public ItemReader<Book> restItCollectReader() {
    return new ItemReader<Book>() {
        @Override
        public Book read() {
                Book nextCollect = null; //ItemReader는 반복문으로 동작한다. 하나씩
Writer로 전달해야 한다.

                if (nextIndex < collectData.size()) {
                    //전체 리스트에서 하나씩 추출해서, 하나씩 Writer로 전달
                    nextCollect = collectData.get(nextIndex);
                    nextIndex++;
                }
                return nextCollect;

        }
    };
}
```

Spring Batch

- Open API 데이터를 읽어서 데이터베이스에 저장하는 배치 프로젝트
  - ✓ Batch 클래스 생성

```
@Bean
public JpaItemWriter<Book> jpaItemWriter(EntityManagerFactory
entityManagerFactory) {

    JpaItemWriter<Book> jpaItemWriter = new JpaItemWriter<>();
    jpaItemWriter.setEntityManagerFactory(entityManagerFactory);
    return jpaItemWriter;
}
```

Spring Batch

- Open API 데이터를 읽어서 데이터베이스에 저장하는 배치 프로젝트
  - ✓ Batch 클래스 생성

```
@Bean
public Step jpaStep(JobRepository jobRepository,
                    PlatformTransactionManager transactionManager,
JpaItemWriter<Book> writer) {
        return new StepBuilder("jpaStep", jobRepository)
                .<Book, Book> chunk(1, transactionManager)
                .reader(restItCollectReader())
                .writer(writer)
                .build();
    }
```

Spring Batch

- Open API 데이터를 읽어서 데이터베이스에 저장하는 배치 프로젝트
  - ✓ Batch 클래스 생성

```
@Bean
public Job jpaJob(JobRepository jobRepository,
                  JobCompletionNotificationListener listener, Step jpaStep) {
    return new JobBuilder("jpaJob", jobRepository)
            .incrementer(new RunIdIncrementer())
            .listener(listener)
            .flow(jpaStep)
            .end()
            .build();
    }

}
```

Spring Batch

- Open API 데이터를 읽어서 데이터베이스에 저장하는 배치 프로젝트
  - ✓ Batch 작업 후 호출되는 리스너 클래스 생성

```
@Component
public class JobCompletionNotificationListener implements JobExecutionListener {
	private static final Logger log =
LoggerFactory.getLogger(JobCompletionNotificationListener.class);
	private final JdbcTemplate jdbcTemplate;
	@Autowired
	public JobCompletionNotificationListener(JdbcTemplate jdbcTemplate) {
		this.jdbcTemplate = jdbcTemplate;
	}
	@Override
	public void afterJob(JobExecution jobExecution) {
		if(jobExecution.getStatus() == BatchStatus.COMPLETED) {
			log.info("!!! JOB FINISHED! Time to verify the results");

			jdbcTemplate.query("SELECT title, price FROM book",
				(rs, row) ->
Book.builder().title(rs.getString(1)).price(rs.getInt(2)).build()
			).forEach(person -> log.info("Found <{{}}> in the
database.", person));
		}
	}
}
```