

```
@Repository
public class GuestbookDao {
    private NamedParameterJdbcTemplate jdbc;
    private SimpleJdbcInsert insertAction;
    private RowMapper<Guestbook> rowMapper = BeanPropertyRowMapper.newInstance(Guestbook.class);

    public GuestbookDao(DataSource dataSource) {
        this.jdbc = new NamedParameterJdbcTemplate(dataSource);
        this.insertAction = new SimpleJdbcInsert(dataSource)
            .withTableName("guestbook")
            .usingGeneratedKeyColumns("id") ;
    }

    public List<Guestbook> selectAll(Integer start, Integer limit) {
        Map<String, Integer> params = new HashMap<>();
        params.put("start", start);
        params.put("limit", limit);
        return jdbc.query(SELECT_PAGING, params, rowMapper);
    }

    public Long insert(Guestbook guestbook) {
        SqlParameterSource params = new BeanPropertySqlParameterSource(guestbook);
        return insertAction.executeAndReturnKey(params).longValue();
    }

    public int deleteById(Long id) {
        Map<String, ?> params = Collections.singletonMap("id", id);
        return jdbc.update(DELETE_BY_ID, params);
    }

    public int selectCount() {
        return jdbc.queryForObject(SELECT_COUNT, Collections.emptyMap(), Integer.class);
    }
}
```

```
private NamedParameterJdbcTemplate jdbc;
```

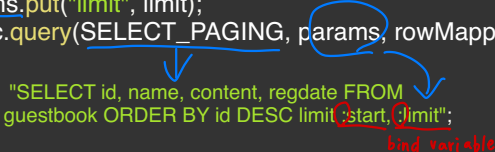
NamedParameterJdbcTemplate는 전통적인 JDBC의 "?" 플레이스홀더 대신에 이름있는(named) 파라미터를 제공하기 위해서 JdbcTemplate를 감싼다. 이 접근방법은 더 좋은 문서화를 제공하고 SQL 문에 다중 파라미터가 있을 때 사용하기가 쉽다

```
private SimpleJdbcInsert insertAction
```

SimpleJdbcInsert와 **SimpleJdbcCall**는 필요한 설정의 양을 제한하려고 데이터베이스의 메타데이터를 최적화한다. 이 접근방법은 코딩을 간소화해서 테이블이나 프로시저의 이름을 제공하고 컬럼명과 일치하는 파라미터들의 맵을 제공하기만 하면 된다. 이는 데이터베이스가 충분한 메타데이터를 제공할 때만 동작한다. 데이터베이스가 이러한 메타데이터를 제공하지 않는다면 파라미터의 명시적인 설정을 제공해야 할 것이다.

```
public GuestbookDao(DataSource dataSource) {  
    this.jdbc = new NamedParameterJdbcTemplate(dataSource);  
    this.insertAction = new SimpleJdbcInsert(dataSource)  
        .withTableName("guestbook")  
        .usingGeneratedKeyColumns("id") ;  
}
```

```
public List<Guestbook> selectAll(Integer start, Integer limit) {  
    Map<String, Integer> params = new HashMap<>();  
    params.put("start", start);  
    params.put("limit", limit);  
    return jdbc.query(SELECT_PAGING, params, rowMapper);  
}
```



"SELECT id, name, content, regdate FROM
guestbook ORDER BY id DESC limit :start, :limit";
bind variable

```
public Long insert(Guestbook guestbook) {
    SqlParameterSource params = new BeanPropertySqlParameterSource(guestbook);
    return insertAction.executeAndReturnKey(params).longValue();
}
```

자바 빈
→ 자동 Mapping

* SQL에 변수 넣는법

① Map으로 할당하기 : 변수로 사용 → SELECT-ALL

② SqlParameterSource 사용 → INSERT

↳ 인터페이스

∴ 구현한 Class인 BeanPropertySqlParameterSource 사용