Spring DB 연동

박태정 [akasha.park@gmail.com]

spring

Spring DB 연동

- 스프링은 다양한 데이터 액세스 기술들과의 통합에 필요한 데이터 액세스용 프레임워크 집합을 제공
  - 저수준(low-level)의 데이터 액세스 관련 작업은 스프링에게 맡기고, 개발자는 애플리케이션의 데이터 관리 로직에만 집중할 수 있다.
  - 데이터 액세스 계층은 퍼시스턴스 기술에 독립적인 방식으로 액세스 할 수 이다.
  - 유연한 애플리케이션 설계를 가능하게 하고, 퍼시스턴스 프레임워크 교체를 가능하게 한다.

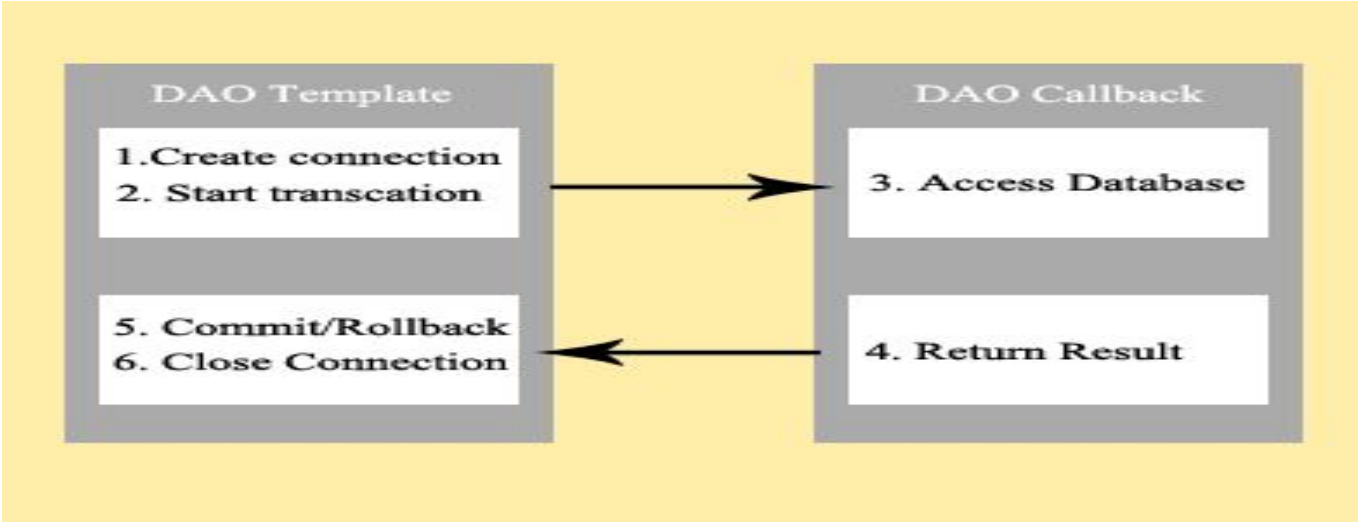
- Data Access Object
  - 데이터베이스에서 데이터를 읽거나 쓰는 수단을 제공하기 위해 존재하며, 이러한 수단은 반드시 인터페이스를 통해 외부에 제공돼야 한다.

Spring DB 연동

- 스프링은 모든 DAO 프레임워크에 걸쳐 일관성 있게 사용할 수 있는 예외 계층구조를 제공한다.
  - 예외가 발생한 문제 상황을 잘 설명하는 데이터 액세스 예외들을 제공
  - 스프링 제공 예외 계층은 어떤 퍼시스턴스 솔루션과도 연관성을 갖지 않으므로 퍼시스턴스 기술과 관계없이 일관성 있게 예외를 발생시킬 수 있다.
  - 스프링 제공 예외 클래스는 모두 DataAccessException을 확장하며 unchecked exception이다.
  - 모든 DataAccess 예외는 반드시 잡아서 처리 할 필요가 없다.
- 스프링의 데이터 액세스 예외 계층의 장점을 취하기 위해서는 반드시 스프링이 제공하는 데이터 액세스 템플릿을 사용해야 한다.

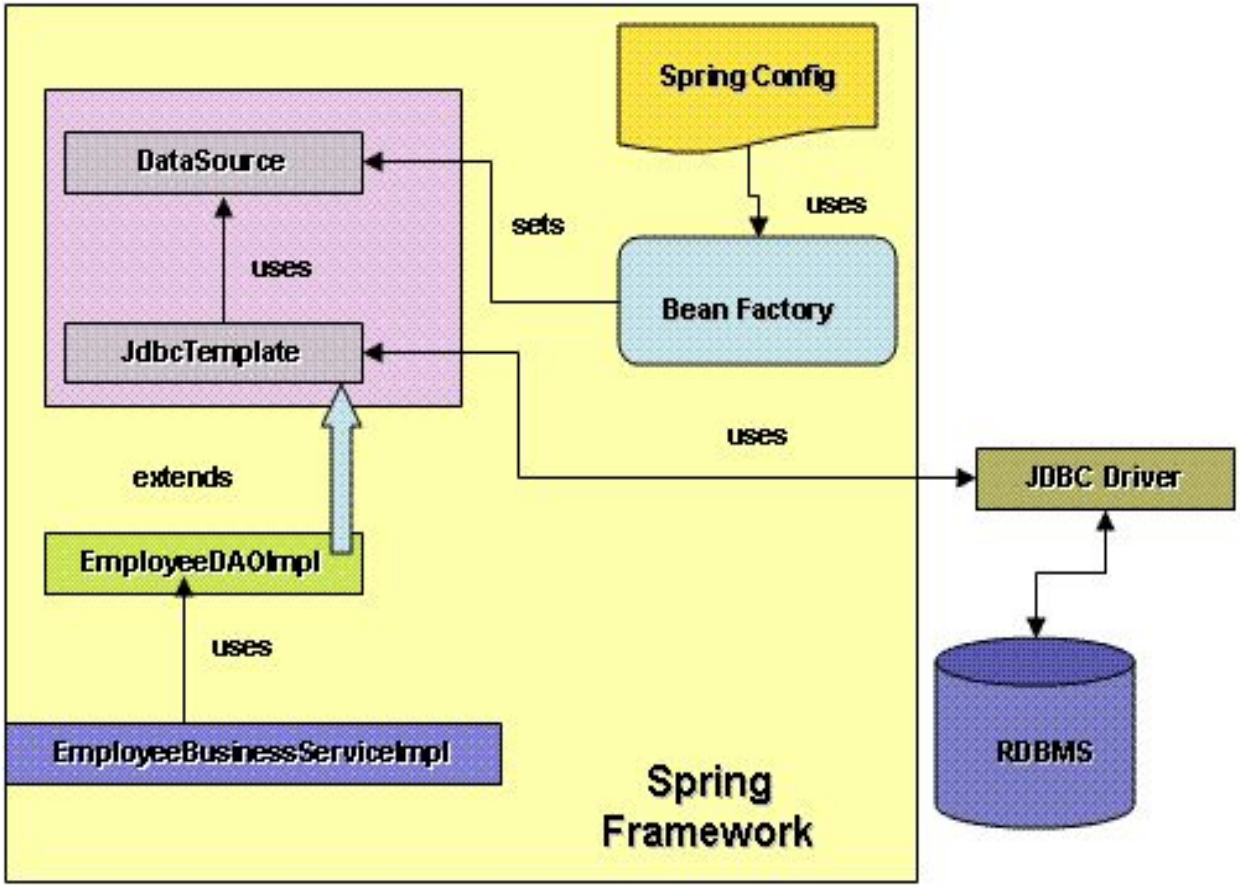
Spring DB 연동

- 템플릿 메소드는 어떤 절차의 골격을 정의한다.
- 스프링은 데이터 액세스 절차상에서 고정된 단계와 가변적인 단계를 template과 callback이라는 두 가지의 별도 클래스로 분리한다..
- 스프링의 템플릿 클래스는 트랜잭션 제어, 자원 관리 및 예외 처리와 같은 데이터 액세스의 고정된 부분을 담당한다.
- 질의 객체(statement) 생성이나 파라미터 바인딩, 질의 결과 추출과 변환 등의 내용은 콜백을 구현해서 처리한다.



박태정[akasha.park@gmail.com]

Spring DB 연동



박태정 [akasha.park@gmail.com]

Spring DB 연동

- 템플릿 클래스를 통한 데이터 접근 지원
  - JdbcTemplate클래스를 사용하면 try-catch-finally 블록 및 커넥션 관리를 위한 중복되는 코드를 줄이거나 없앨 수 있다.
  - JdbcTemplate 클래스는 처리과정에서 SQLException이 발생하면 스프링이 제공하는 예외 클래스로 변환해서 발생시킨다
- DataSource설정
  - 커넥션 풀을 이용한 DataSource설정
    - DBCP API와 같은 커넥션 풀 라이브러리를 이용
  - JNDI를 이용한 DataSource 설정
  - DriverManager를 이용한 DataSource 설정

Spring DB 연동

- 커넥션 풀을 이용한 DataSource설정

```
<!-- BasicDataSource DBCP를 이용한 설정 예시  -->
<bean id="dataSource"
            class="org.apache.commons.dbcp.BasicDataSource"
            p:driverClassName="oracle.jdbc.OracleDriver"
            p:url="jdbc:oracle:thin:@127.0.0.1:1521:orcl"
            p:username="hr" p:password="hr"
           destroy-method="close" />
```

Spring DB 연동

- BasicDataSource

프로퍼티	설명
initialSize	초기에 생성될 커넥션개수.
maxActive	커넥션풀이 제공할 최대 커넥션개수
maxIdle	사용되지 않고 풀에 저장될수 있는 최대 커넥션개수, 음수일경우 제한없음.
minIdle	사용되지 않고 풀에 저장될수 있는 최소 커넥션개수.
maxWait	풀에 커넥션이 존재하지 않을때, 커넥션이 풀에 다시 리턴되기까지 대기시간. 1/1000초단위, -1일경우 무한대기.
minEvictableIdleTimeMillis	사용되지 않은 커넥션을 추출할 때, 이 속성에서 지정한 시간이상 비활성화 상태인 커넥션만 추출한다. 양수가 아닌 경우 비활성화 된 시간으로는 풀에서 제거 안됨.

Spring DB 연동

- BasicDataSource

프로퍼티	설명
timeBetweenEvictionRunsMillis	사용되지 않은 커넥션을 추출하는 쓰레드의 실행주기를 설정. 양수가 아닌 경우 실행되지 않음.
numTestsPerEvictionRun	사용되지 않은 커넥션을 몇 개 검사할지 지정.
testOnBorrow	true일 경우, 커넥션풀에서 커넥션을 가져올 경우, 커넥션이 유효한지 검사.
testOnReturn	true일 경우, 커넥션풀에 커넥션을 반환할 때, 커넥션이 유효한지 검사.
testWhileIdle	true일 경우, 비활성화 커넥션을 추출할 때, 커넥션이 유효한지 검사해서 유효하지 않은 커넥션은 풀에서 제거.

Spring DB 연동

- 커넥션 풀 API c3p0 라이브러리를 이용한 DataSource설정

```
<!-- BasicDataSource DBCP를 이용한 설정 예시  -->
<bean id="dataSource"
            class="com.mchange.v2.c3p0.ComboPooledDataSource"
            p:driverClassName="oracle.jdbc.OracleDriver"
            p:url="jdbc:oracle:thin:@127.0.0.1:1521:orcl"
            p:username="hr" p:password="hr"
           destroy-method="close" />
```

Spring DB 연동

- JNDI를 이용한 DataSource설정
  - /WEB-INF/dataSource.xml 설정DataSource설정

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:p="http://www.springframework.org/schema/p"
        xmlns:jee="http://www.springframework.org/schema/jee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
    http://www.springframework.org/schema/jee
    http://www.springframework.org/schema/jee/spring-jee-3.0.xsd">

        <!-- JNDI 기반의 설정 설정 예시 -->
        <jee:jndi-lookup id="dataSource" jndi-name="jdbc/oracle"
                resource-ref="true" />
</beans>
```

Spring DB 연동

- JndiObjectFactoryBean를 이용 JNDI로부터 DataSource 얻기
  - /WEB-INF/dataSource.xml 설정DataSource설정

```
<!-- JNDI 기반의 설정 설정 예시 -->
<bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean">
            <property name="jndiName" value="jdbc/oracle" />
            <property name="resourceRef" value="true" />
</bean>
</beans>
```

Spring DB 연동

- DriverManager를 이용한 DataSource설정
  - 애플리케이션이 커넥션을 요청할 때마다 새로운 커넥션을 반환
  - DriverManagerDataSource가 공급하는 커넥션은 풀링되지 않는다.

```
 <!-- DriverManagerDataSource를 이용한 설정 예시 -->
<bean id="dataSource"
        class="org.springframework.jdbc.datasource.DriverManagerDataSource"
           p:driverClassName="oracle.jdbc.OracleDriver"
           p:url="jdbc:oracle:thin:@127.0.0.1:1521:orcl"
           p:username="hr" p:password="hr" />
</beans>
```

- SingleConnectionDataSource
  - 커넥션을 요청하면 항상 동일한 커넥션을 반환
  - 커넥션은 풀링되지 않는다.

Spring DB 연동

- DataSourceUtils
  - 스프링이 제공하는 트랜잭션 기능을 올바르게 사용하고 싶다면 스프링이 제공하는 DataSourceUi신 클래스를 이용해서 Connectio을 구하고 반환해야 한다.

```
Connection conn = null;
try{
    conn = DataSourceUtils.getConnection(dataSource);
    ...
  }finally{
      ...
      DataSourceUtils.releaseConnection(conn, dataSource);
  }
```

JDBC지원 클래스

- JdbcTemplate
  - 기본적인 JDBC 템플릿 클래스로서 JDBC를 이용해서 데이터에 대한 접근을 제공
  - Indexed parameter 기반의 쿼리를 통해 데이터베이스를 쉽게 액세스하는 기능을 제공

- NamedParameterJdbcTemplate
  - PreparedStatement에서 인덱스 기반의 파라미터가 아닌 이름을 가진 파라미터를 사용할 수 있도록 지원하는 템플릿 클래스

- SimpleJdbcTemplate
  - 자바 5의 variable parameter list, generics, autoboxing등을 이용해서 쿼리를 실행할때 사용되는 데이터를 전달할 수 있는 템플릿 클래스
  - 2.5버전에서 NamedParameterJdbcTemplate 기능이 합쳐졌다

JDBC지원 클래스

- SimpleJdbcInsert
  - 데이터 삽입을 위한 인터페이스를 제공해주는 클래스
- SimpleJdbcCall
  - 프로시저 호출을 위한 인터페이스를 제공해주는 클래스

Spring DB 연동

- 조회 메서드
  - 쿼리 실행 결과를 객체 목록으로 가져올 때는 RowMapper를 이용하는 query()를 이용
    - query(String sql, RowMapper<T> rowMapper)
    - List<T> query(String sql, Object[] args, RowMapper<T> rowMapper)
    - List<T> queryquery(String sql, Object[] args, int[] argTypes, RowMapper<T> rowMapper)
      - RowMapper는 실행 결과를 자바 객체로 변환환해주는 매퍼
      - args 파라미터는PreparedStatement를 실행할 때 사용할 파라미터 바인딩 값 목록
      - argTypes는 파라미터 바인딩을 할때 사용할 SQL 타입 목록

Spring DB 연동

- 조회 메서드
  - 쿼리 실행 결과로 읽어온 컬럼 개수가 한 개인 경우
    - List<T> queryForList(String sql, Class<T> elementType)
    - List<T> queryForList (String sql, Object[] args, Class<T> elementType)
    - List<T> queryForList(String sql, Object[] args, int[] argTypes, Class<T> elementType)
      - args 파라미터는PreparedStatement를 실행할 때 사용할 파라미터 바인딩 값 목록
      - argTypes는 파라미터 바인딩을 할때 사용할 SQL 타입 목록

Spring DB 연동

- 조회 메서드
  - 쿼리 실행 결과 행의 개수가 한 개인 경우
    - T queryForObject(String sql, RowMapper<T> rowMapper)
    - T queryForObject(String sql, Object[] args, RowMapper<T> rowMapper)
    - T queryForObject(String sql, Object[] args, int[] argTypes, RowMapper<T> rowMapper)
    - T queryForObject (String sql, Class<T> requiredType)
    - T queryForObject (String sql, Object[] args , Class<T> requiredType)
    - T queryForObject (String sql, Object[] args , int[] argTypes, Class<T> requiredType)
  - 행의 개수가 0이거나 2개 이상인 경우에는 IncorrectResultSizeDataAccessException 예외 발생

Spring DB 연동

- 조회 메서드
  - 쿼리 실행 결과 Object가 아닌 int나 long 타입의 결과를 구할 때
    - int queryForInt(String sql)
    - int queryForInt(String sql, Object[] args,)
    - int queryForInt(String sql, Object[] args, int[] argTypes)
    - long queryForLong(String sql)
    - long queryForLong(String sql, Object[] args,)
    - long queryForLong(String sql, Object[] args, int[] argTypes)

  - 행의 개수가 0이거나 2개 이상인 경우에는 IncorrectResultSizeDataAccessException 예외 발생

Spring DB 연동

- Insert, update, delete 수행 메서드
  - 쿼리 실행 결과 변경된 행의 개수 리턴
    - int update(String sql)
    - int update (String sql, Object[] args,)
    - int update (String sql, Object[] args, int[] argTypes)

Spring DB 연동

- NamedParameterJdbcTemplate
  - 이름 기반의 파라미터를 설정할 수 있도록 해준다.

```
public class NamedJdbcTemplateMessageDao implements MessageDao {

    private NamedParameterJdbcTemplate jdbcTemplate;

    public void setJdbcTemplate(NamedParameterJdbcTemplate jdbcTemplate) {
                        this.jdbcTemplate = jdbcTemplate;
            }
 ....
}
```

```
private static final String INSERT_SQL =
"insert into GUESTBOOK (GNAME, CONTENT) values (:guestName, :content)";
```

Spring DB 연동

- NamedParameterJdbcTemplate
  - 파라미터 값을 설정하기 위해 Map이나 SqlParameterSource를 전달 받는다.

```
public int delete(int id) {
    Map<String, Object> paramMap = new HashMap<String, Object>();
    paramMap.put("id", id);
    return template.update("delete from GUESTBOOK where GID = :id",
            paramMap);
            }
```

  - 이름 기반의 파라미터를 갖지 않는 쿼리를 실행할 경우에는 아무 값도 갖지 않는 Map 객체를 사용

```
int  id  =  template.queryForInt(“Select last_insert_id() ” ,
                    Collections<String, Object> emptyMap());
```

Spring DB 연동

- NamedParameterJdbcTemplate가 제공하는 Map 기반 메소드
  - List<T> queryForList(String sql, Map<String, ?> paramMap, RowMapper<T> rowMapper)
  - List<T> queryForList (queryForList(String sql, Map<String, ?> paramMap, Class<T> elementType)
  - T queryForObject (String sql, Map<String, ?> paramMap, RowMapper<T> rowMapper)
  - T queryForList (queryForList(String sql, Map<String, ?> paramMap, Class<T> requiredType)
  - int queryForInt (String sql, Map<String, ?> paramMap)
  - long queryForLong (String sql, Map<String, ?> paramMap)
  - int update (String sql, Map<String, ?> paramMap)

Spring DB 연동

- NamedParameterJdbcTemplate가 제공하는 SqlParameterSource 기반 메소드
  - List<T> queryForList(String sql, SqlParameterSource paramSource, RowMapper<T> rowMapper)
  - List<T> queryForList (queryForList(String sql, SqlParameterSource paramSource, Class<T> elementType)
  - T queryForObject (String sql, SqlParameterSource paramSource, RowMapper<T> rowMapper)
  - T queryForList (queryForList(String sql, SqlParameterSource paramSource, Class<T> requiredType)
  - int queryForInt (String sql, SqlParameterSource paramSource)
  - long queryForLong (String sql, SqlParameterSource paramSource)
  - int update (String sql, SqlParameterSource paramSource)
- SqlParameterSource 인터페이스 구현 클래스
  - BeanPropertySqlParameterSource
    - 동일한 이름을 갖는 자바 객체의 프로퍼티 값을 이용해서 파라미터 값을 설정
  - MapSqlParameterSource
    - <이름, 값>쌍을 이용해서 파라미터의 값을 설정

Spring DB 연동

- SimpleJdbcTemplate
  - 이름 기반 파라미터 설정과 인덱스 기반의 파라미터 설정을 모두 지원
- SimpleJdbcInsert
  - 쿼리를 사용하지 않고 데이터를 삽입할 수 있도록 해주는 클래스
  - usingColumns()를 사용하면 쿼리를 생성할 때 사용할 컬럼을 직접 지정
  - execute()를 이용하여 데이터 삽입
    - Int execute(Map<String, Object> args)
    - Int execute(SqlParameterSource parameterSource)
  - 데이터 삽입시 자동으로 생성되는 키 컬럼을 구하고 싶을때 executeAndReturnKey()사용
    - Number executeAndReturnKey(Map<String, Object> args)
    - Number executeAndReturnKey(SqlParameterSource paramSource)
    - KeyHolder executeAndReturnKey(Map<String, Object> args)
    - KeyHolder executeAndReturnKey(SqlParameterSource paramSource)

박태정[akasha.park@gmail.com]