

# Spring Framework

## 18. MyBatis 어플리케이션 작성(2)

# CONTENTS

1

Mapper 인터페이스 개념

2

Mapper 인터페이스 작성 및 설정

3

여러 개의 Mapper 인터페이스 설정

# 학습 목표

- Mapper 인터페이스 개념에 대하여 이해할 수 있습니다.
- Mapper 인터페이스 작성 및 설정에 대하여 이해할 수 있습니다.
- 여러 개의 Mapper 인터페이스 설정에 대하여 이해할 수 있습니다.



## 1. Mapper 인터페이스 개념

## ■ Mapper 인터페이스

Mapper 인터페이스는 Mapping 파일에 기재된 **SQL**을  
**호출하기 위한 인터페이스**

- ◉ Mapper 인터페이스는 SQL을 호출하는 프로그램을 Type Safe 하게 기술하기 위해 MyBatis3.x부터 등장
- ◉ Mapping 파일에 있는 SQL을 자바 인터페이스를 통해 호출할 수 있도록 해줌

## ■ Mapper 인터페이스를 사용하지 않았을 때

```
session.selectOne("userNS.selectUserById", userid);
```

UserDAOImpl  
read(userid)

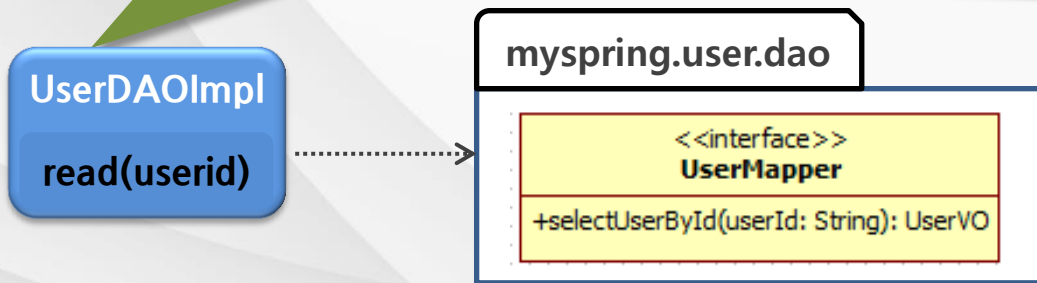
```
<mapper namespace="userNS">  
  <select id="selectUserById" parameterType="String">
```

- Mapper 인터페이스를 사용하지 않으면,  
SQL을 호출하는 프로그램은 SqlSession의 메서드의 아규먼트에 문자열로 **네임스페이스+"."+SQL ID** 로 지정해야 함
- 문자열로 지정하기 때문에 오타에 의해 버그가 숨어있거나,  
IDE에서 제공하는 code assist 를 사용할 수 없음

## Mapper 인터페이스를 사용하였을 때

```
userMapper.selectUserById(userid);
```

```
<mapper namespace="myspring.user.dao.UserMapper">  
  <select id="selectUserById" parameterType="String">
```



- UserMapper 인터페이스는 개발자가 작성
- 패키지 이름+ "." + 인터페이스 이름 + "." + 메서드 이름이 네임스페이스 + "." + SQL ID가 되도록 네임스페이스와 SQL의 ID를 설정해야 함
- 네임스페이스 속성에는 패키지를 포함한 Mapper 인터페이스 이름
- SQL ID에는 매핑하는 메서드 이름을 지정하는 것

A person's hands are shown holding a smartphone, with the screen glowing. The background is dark with out-of-focus, colorful bokeh lights in shades of yellow, orange, and blue. A semi-transparent dark banner with a yellow decorative element on the left is positioned at the bottom of the image.

## 2. Mapper 인터페이스 작성 및 설정



### ■ Mapper 인터페이스 작성

```
UserMapper.java ✕  
1 package myspring.user.dao;  
2  
3 import java.util.List;  
4  
5  
6  
7 public interface UserMapper {  
8     UserVO selectUserById(String id);  
9     List<UserVO> selectUserList();  
10    void insertUser(UserVO userVO);  
11    void updateUser(UserVO userVO);  
12    void deleteUser(String id);  
13 }
```


### ■ Mapping 파일 수정

기  
존

```
<mapper namespace="userNS">
  <select id="selectUserById" parameterType="string" resultType="User">
    select * from users where userid=#{value}
  </select>
```

변  
경

```
<mapper namespace="myspring.user.dao.UserMapper">
  <select id="selectUserById" parameterType="string" resultType="User">
    select * from users where userid=#{value}
  </select>
```



## ■ DAO 클래스 수정

기  
존

```
@Repository("userDao")
public class UserDaoImpl implements UserDao {
    @Autowired
    private SqlSession session;
    @Override
    public UserVO read(String id) {
        UserVO user = session.selectOne("userNS.selectUserById", id);
        return user;
    }
}
```

변  
경

```
@Repository("userDao")
public class UserDaoImpl implements UserDao {
    @Autowired
    private UserMapper userMapper;
    @Override
    public UserVO read(String id) {
        UserVO user = userMapper.selectUserById(id);
        return user;
    }
}
```



### ■ MapperFactoryBean의 설정

```
beans.xml
10      <!-- Mapper 설정 -->
11      <bean id="userMapper" class="org.mybatis.spring.mapper.MapperFactoryBean">
12          <property name="mapperInterface" value="myspring.user.dao.UserMapper" />
13          <property name="sqlSessionTemplate" ref="sqlSession" />
14      </bean>
```

- MapperFactoryBean은 UserMapper를 구현하는 프록시 클래스를 생성하고, 그것을 어플리케이션에 injection 한다.
- 프록시는 런타임 시에 생성되므로, 지정된 Mapper는 실제 구현 클래스가 아닌, 인터페이스여야만 한다.
- MapperFactoryBean은 sqlSessionFactory나 sqlSessionTemplate를 필요로 한다.

### Mapper 인터페이스의 사용 테스트

```
*UserController.java
21 public class UserController {
22     @Autowired
23     ApplicationContext context;
24     @Autowired
25     UserService service;
26
27     @Test
28     public void getUserTest() {
29         UserVO user = service.getUser("gildong");
30         System.out.println("User 정보 : " + user);
31         assertEquals("홍길동", user.getName());
32     }
```

#### 실행 결과

```
Markers Console Maven Repositories JUnit Properties
<terminated> UserController (2) [JUnit] C:\Program Files (x86)\Java\jre1.8.0_91\bin
UserController.read(..) 실행 시간 : 1520 ms
UserService.getUser(..) 실행 시간 : 1521 ms
```

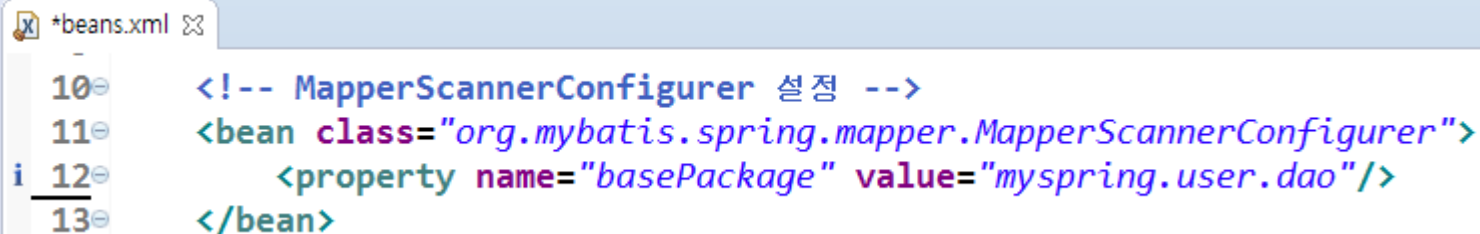
A person's hands are shown holding a smartphone, with the screen displaying a bright, glowing light. The background is dark and filled with out-of-focus, circular bokeh lights in warm yellow and orange tones. A semi-transparent dark banner with a yellow decorative element on the left side is positioned at the bottom of the image.

### 3. 여러 개의 Mapper 인터페이스 설정

#### ■ MapperScannerConfigurer의 사용

- ◉ MapperFactoryBean을 이용해 Mapper 인터페이스를 등록할 때 Mapper 인터페이스의 개수가 많아지게 되면 일일이 정의하는데 시간이 많이 걸림
- ◉ Mapper 인터페이스의 수가 많아지면 MapperScannerConfigurer를 이용하여 Mapper 인터페이스의 객체를 한 번에 등록하는 것이 편리함
- ◉ MapperScannerConfigurer를 이용하면 지정한 패키지 아래 모든 인터페이스가 Mapper 인터페이스로 간주되어 Mapper 인터페이스의 객체가 DI 컨테이너에 등록되는 것

#### ■ MapperScannerConfigurer의 설정



```
*beans.xml
10  <!-- MapperScannerConfigurer 설정 -->
11  <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
i 12      <property name="basePackage" value="myspring.user.dao"/>
13  </bean>
```

- ◉ basePackage 속성에서 지정하는 것은 Mapper 인터페이스를 검색할 대상이 되는 Package
- ◉ myspring.user.dao 아래의 인터페이스들은 모두 Mapper 인터페이스에 대응하여 Mapper 객체가 생성된다는 점
- ◉ 예상하지 않은 다른 객체가 등록되어 오류가 발생할 수 있음



#### ■ MapperScannerConfigurer의 설정 테스트

```
*UserController.java
21 public class UserController {
22     @Autowired
23     ApplicationContext context;
24     @Autowired
25     UserService service;
26
27     @Test
28     public void getUserTest() {
29         UserVO user = service.getUser("gildong");
30         System.out.println("User 정보 : " + user);
31         assertEquals("홍길동", user.getName());
32     }
```

#### 실행 결과

Caused by:  
[org.springframework.context.annotation.ConflictingBeanDefinitionException](#): Annotation-specified bean name 'userDao' for bean class [myspring.user.dao.UserDao] conflicts with existing, non-compatible bean definition of same name and class [myspring.user.dao.UserDaoImpl]

#### ■ Marker 인터페이스와 Marker 어노테이션의 사용

- 검색의 대상이 되는 Package 아래의 인터페이스들 중에서 Mapper로서 작성한 인터페이스로만 범위를 좁히려면 Marker 인터페이스와 Marker 어노테이션을 작성하여 MapperScannerConfigurer에 설정하면 됨

##### Marker 인터페이스

```
*MyMapper.java
1 package myspring.user.dao;
2
3 public @interface MyMapper {
4 }
```

##### Marker 어노테이션 부여

```
*UserMapper.java
1 package myspring.user.dao;
2+ import java.util.List;
3
4 @MyMapper
5 public interface UserMapper {
6     UserVO selectUserById(String id);
7     List<UserVO> selectUserList();
8     void insertUser(UserVO userVO);
9     void updateUser(UserVO userVO);
10    void deleteUser(String id);
11 }
```

#### ■ MapperScannerConfigurer에 Marker 어노테이션 지정

```
beans.xml
10      <!-- MapperScannerConfigurer 설정 -->
11      <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
12          <property name="basePackage" value="myspring.user.dao"/>
13          <property name="annotationClass" value="myspring.user.dao.MyMapper" />
14      </bean>
```

- ◉ org.mybatis.spring.mapper.MapperScannerConfigurer :  
컴포넌트 스캔을 통하여 Mapper를 찾기 위한 설정
- ◉ basePackage : Mapper를 찾는 베이스 패키지
- ◉ annotationClass : Mapper를 지정하는 어노테이션 클래스

#### ■ MapperScannerConfigurer의 설정 테스트

```
*UserClient.java
21 public class UserClient {
22     @Autowired
23     ApplicationContext context;
24     @Autowired
25     UserService service;
26
27     @Test
28     public void getUserTest() {
29         UserVO user = service.getUser("gildong");
30         System.out.println("User 정보 : " + user);
31         assertEquals("홍길동", user.getName());
32     }
```

#### 실행 결과

```
Markers Console Maven Repositories JUnit Properties
<terminated> UserClient (2) [JUnit] C:\Program Files (x86)\Java\jre1.8.0_91\bin
UserDao.read(...) 실행 시간 : 1520 ms
UserService.getUser(..) 실행 시간 : 1521 ms
```



학습정리

지금까지 **[MyBatis 어플리케이션 작성(2)]**에 대해서 살펴보았습니다.

## Mapper 인터페이스 개념

- ◉ Mapping 파일의 SQL문 호출을 Type Safe 하게 호출하기 위한 인터페이스

## Mapper 인터페이스 작성 및 설정

- ◉ Mapper 인터페이스 작성
- ◉ MapperFactoryBean을 이용한 Mapper 등록

## 여러 개의 Mapper 인터페이스 설정

- ◉ 여러 개의 Mapper 인터페이스 작성
- ◉ MapperScannerConfigurer을 이용한 Mapper 등록