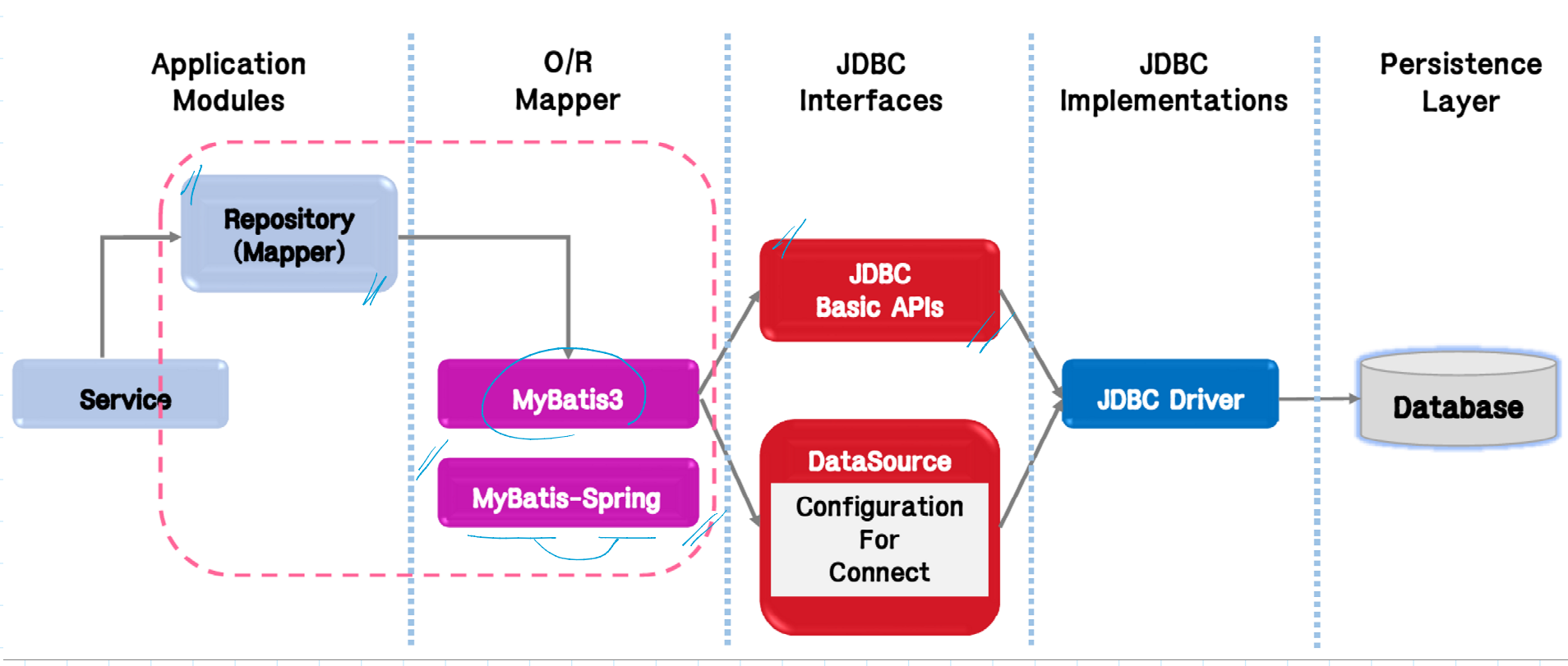
1023

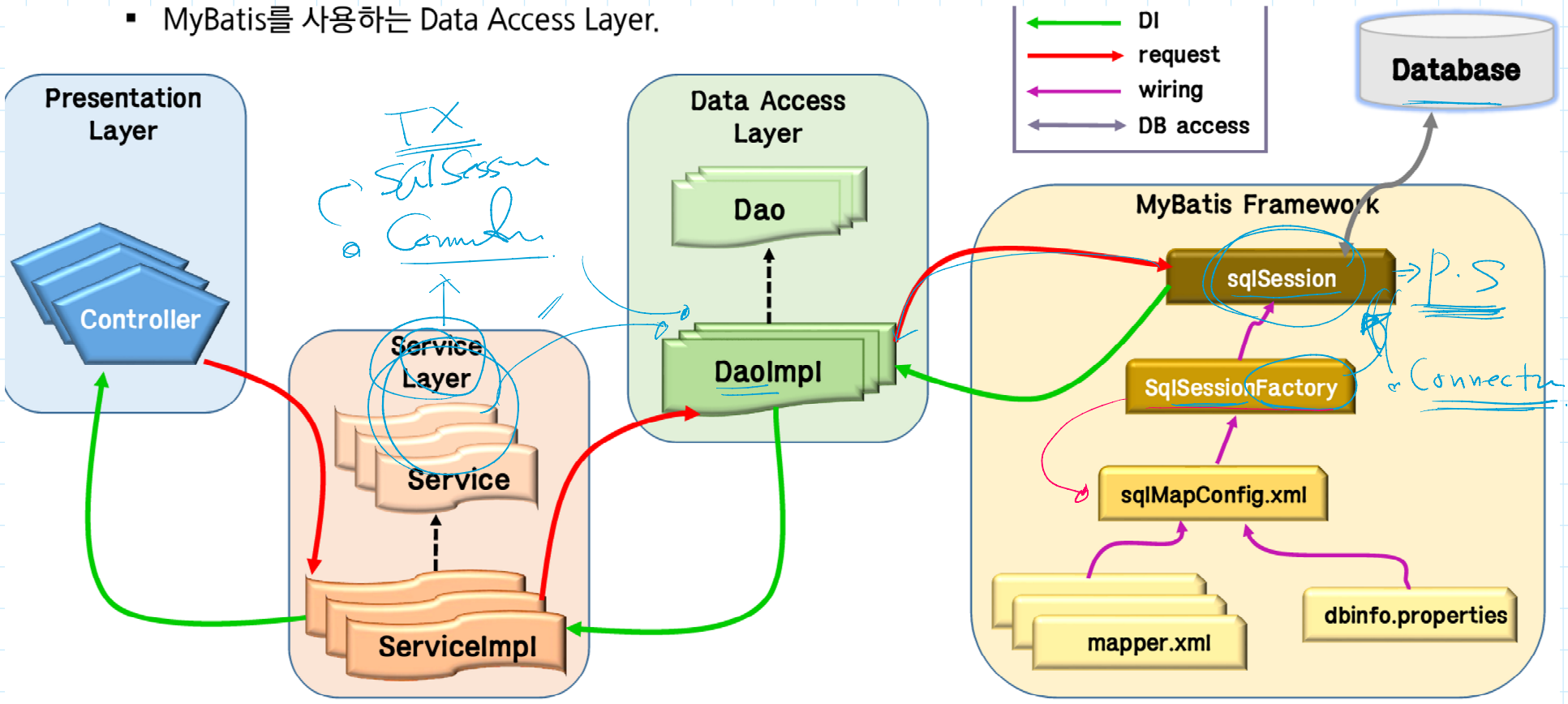
2020년 10월 23일 금요일 오전 6:01

MyBatis Spring: https://mybatis.org/spring/ko/

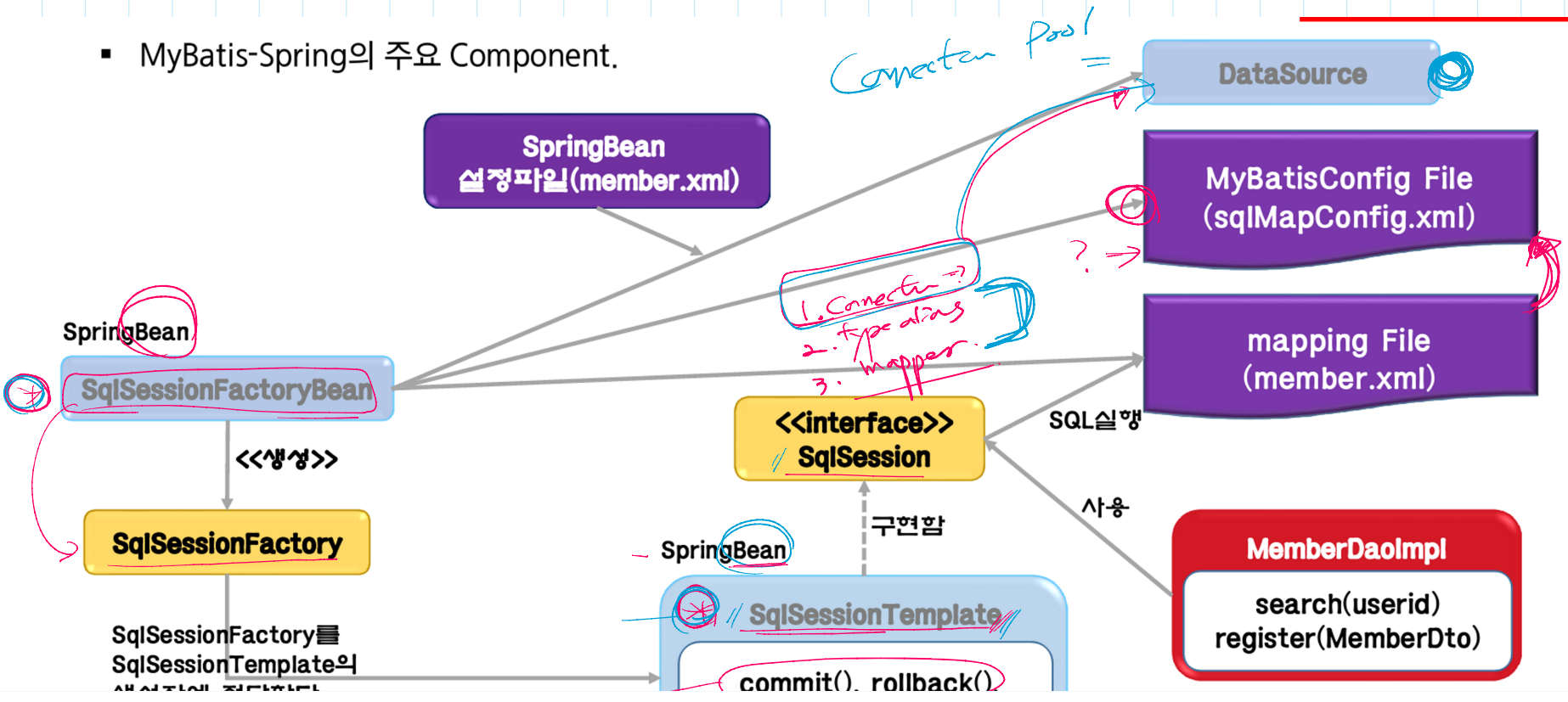
MyBatis 관련: https://goodteacher.tistory.com/246?category=824749

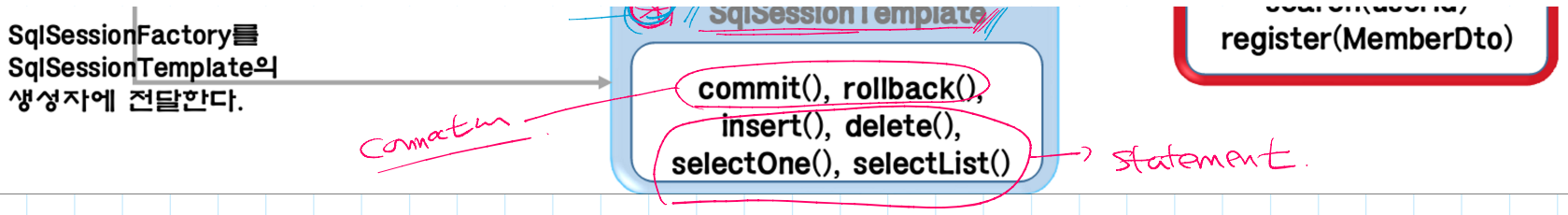


- MyBatis를 사용하는 Data Access Layer.



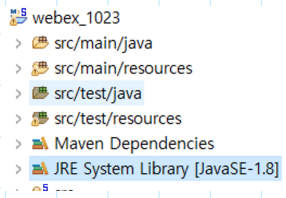
- MyBatis-Spring의 주요 Component.





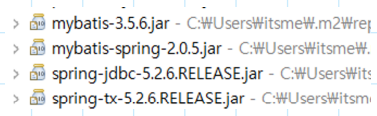
회원관리를 위한 프로젝트를 진행해보자.
- 회원의 등록/수정/삭제/조회/목록조회/다양한조회를 할수 있다.
- 이를 위해 Spring과 MyBatis framework를 사용한다.

1. 프로젝트 생성
   a. 버전 맞추기는 기본
   b.

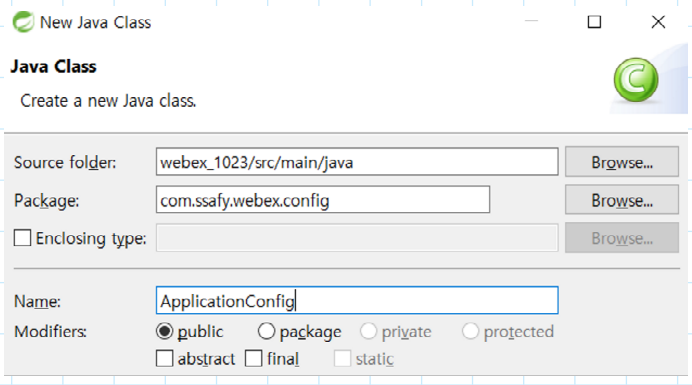


2. 필요한 라이브러리들은 어떻게 될까?
   a. spring, spring web = mvc project이기 때문에 이미 완료됨
   b. DB연결: mybatis, mysql, spring-jdbc --> mvnrepository.com

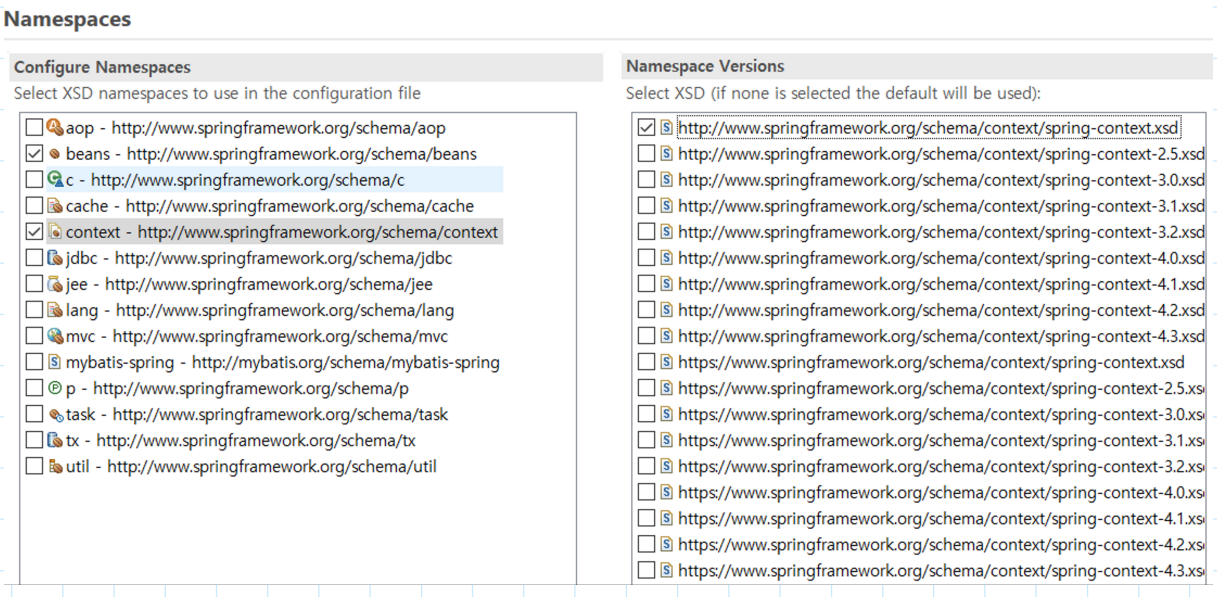
```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.22</version>
</dependency>

<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.5.6</version>
</dependency>
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>2.0.5</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
```



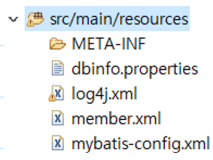
3. Project 수정 - javaconfig 방식으로 바꿔보자..
   i. Application-config.xml 등을 --> Java config로 변경
   ii. 설정 파일 2개를 만든다.
   iii.



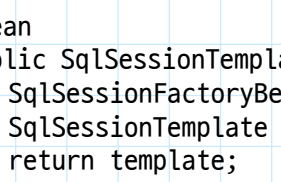
   iv. Xml에서 위 설정 파일을 호출하도록 변경해보자.
   v.



   vi. Xml config의 내용을 자바 기반으로 바꿔보자.
4. Mybatis관련 환경 설정을 정리해보자.
   i. Mybatis config와 mapper

ii. 

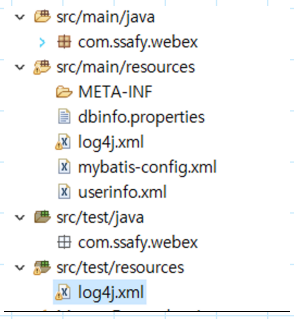
iii. 수정할 부분이 있는지 찾아보자.

iv. 

5. 설정 끝....
6. 이제 필요한 빈들을 만들어볼까..
   i. SqlSessionTemplate, SqlSessionFactoryBean, Datasource 빈을 만들자.
   ii. 이녀석들은 웹과 관련이 있나?? --> ApplicationConfig

```
@Bean
public DataSource ds() {
    DriverManagerDataSource ds = new DriverManagerDataSource();
    ds.setDriverClassName("com.mysql.cj.jdbc.Driver");
    // 이시점에서 확인할 것들은??? schema & 계정 정보
    ds.setUrl("jdbc:mysql://localhost:3306/springWork?serverTimezone=UTC&useUniCode=yes&characterEncoding=UTF-8");
    ds.setUsername("ssafy");
    ds.setPassword("ssafy");
    return ds;
}
```

```
@Bean
public SqlSessionFactoryBean factoryBean() {
    SqlSessionFactoryBean factory = new SqlSessionFactoryBean();
    factory.setDataSource(ds());
    factory.setConfigLocation(new ClassPathResource("mybatis-config.xml"));
    return factory;
}


@Bean
public SqlSessionTemplate sqltemplate() throws Exception {
    SqlSessionFactoryBean factory = factoryBean();
    SqlSessionTemplate template = new SqlSessionTemplate(factory.getObject());
    return template;
}
```

7. 우리는 잘 한걸까?? 테스트 해보자.
   i. 개발과 테스트는 분리해서 관리한다.
   ii. 


```
public class TestClient {
    public static void main(String[] args) {
        ApplicationContext ctx = new AnnotationConfigApplicationContext(ApplicationConfig.class);
        String[] beans = ctx.getBeanDefinitionNames();
        for (String bean : beans) {
            System.out.println(bean);
        }
    }
}
```

위와 같은 테스트가 여러 번 진행된다면?? --> 단위테스트 도입 --> 테스트의 자동화
스프링에서 단위테스트를 하기 위해서는 spring-test와 junit(4.12 버전 이상) 필요

```
<!-- Test -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.1</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>${org.springframework-version}</version>
    <scope>test</scope>
```

```
</dependency>
```

단위테스트 기본 코드

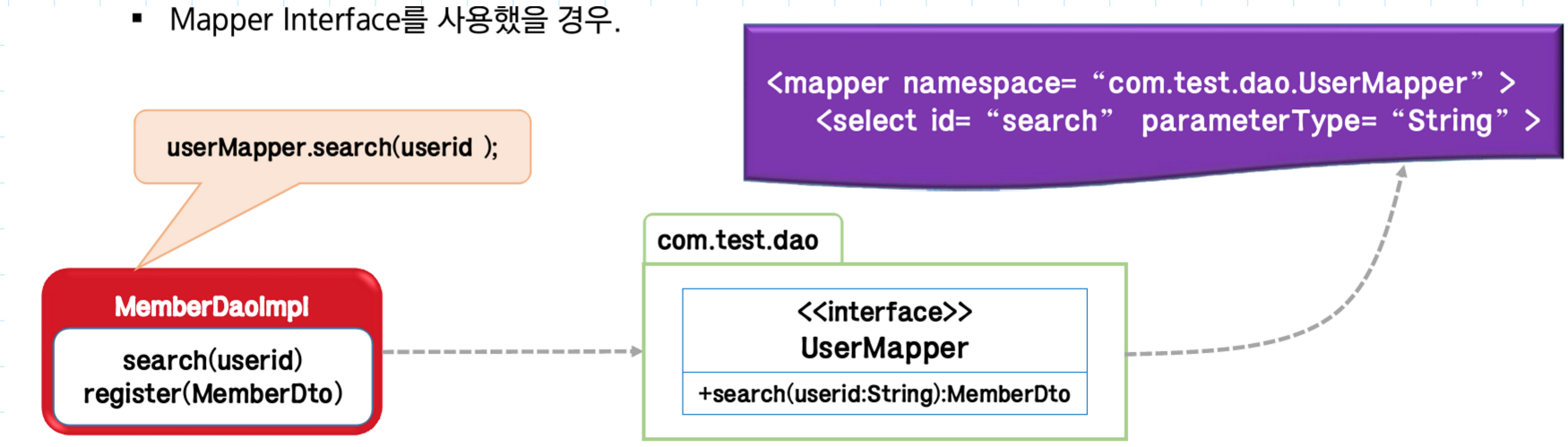
```
// 단위테스트를 스프링과 연동하겠습니다.
@RunWith(SpringRunner.class)
// 환경설정은 아래꺼를 이용해주세요.
@ContextConfiguration(classes = ApplicationConfig.class)
public class BeanTest {

   private static final Logger logger = LoggerFactory.getLogger(BeanTest.class);

   @Autowired
   DataSource ds;

   @Autowired
   SqlSessionTemplate st;

   @Test // 모든 테스트들은 @Test를 갖는다.
   public void dsTest() {
      //logger.debug("ds: {}" , ds);
      assertNotNull(ds);
   }

      @Test
      public void templateTest() {
         assertNotNull(st);
      }

   }
```

드디어 비즈니스 로직인가....

DTO > REPO > mapper > Test > Service > Test --> model 완성 > view 작성 > Controller 작성 > 전체 테스트

1. DTO 생성
2. REPO 생성
   - Mapper Interface를 사용했을 경우.

a.



3. Mapper에 메서드 작성
4. 단위테스트
5. Service 작성
6. Service에 대한 단위테스트
7. 화면 만들기
8. 컨트롤러 연결해주기

DML 처리

```
// 스프링아.. 트랜젝션을 부탁해~~
@Bean
public PlatformTransactionManager transactionManager() {
   return new DataSourceTransactionManager(ds());
}
```

```
@Test
```

```
@Transactional// 테스트 끝난 후 자동 rollback
public void insertTest() {
    UserInfo user = new UserInfo("hong", "홍길동", "hong");
    int result = urepo.insert(user);
    assertTrue(result==1);

    UserInfo selected = urepo.select(user.getUserId());
    assertEquals(user.getName(), selected.getName());
}


@Override
@Transactional// 메서드 내에서 runtime exception 발생 --> rollback, 아니면 commit
public int join(UserInfo user) {
    int result = urepo.insert(user);
    return result;
}
```

Controller에 전달된 예외 처리하기

```
@ExceptionHandler
// ExceptionHandler 메서드는 Model을 받을 수가 없어요. ㅜㅜ
public ModelAndView exHandler(Exception e) {
    ModelAndView mv = new ModelAndView();
    mv.addObject("error", e.getMessage());
    mv.setViewName("error/500");
    return mv;
}
```

좀 더 보면 좋을것들: 동적 쿼리