MyBatis 2
- Mapper -

❖ Member 테이블 정의

```
CREATE TABLE MEMBER(
    USER_ID         VARCHAR2(20) PRIMARY KEY,
    NAME            VARCHAR2(20 CHAR) NOT NULL,
    PASSWORD        VARCHAR2(200) NOT NULL,
    USER_LEVEL      VARCHAR2(10) NOT NULL
       CHECK ( USER_LEVEL IN (
                    'NORMAL', 'SILVER', 'GOLD', 'ADMIN')),
    PHONE_NUMBER    VARCHAR2(20),
    NICK_NAME       VARCHAR2(20 CHAR),
    EMAIL           VARCHAR2(30) NOT NULL,
    ADDRESS         VARCHAR2(200 CHAR),
    REG_DATE        DATE DEFAULT SYSDATE,
    UPDATE_DATE     DATE DEFAULT SYSDATE
);
```

데이터베이스 준비

❖ Member 테이블 테스트 데이터 준비

```
INSERT INTO MEMBER (
   USER_ID, NAME, PASSWORD, USER_LEVEL,
   PHONE_NUMBER, NICK_NAME,
   EMAIL, REG_DATE, UPDATE_DATE)
SELECT
      LOWER(EMAIL), LAST_NAME, '1234', 'NORMAL',
      REPLACE(PHONE_NUMBER, '.', '-'), LAST_NAME,
      LOWER(EMAIL) || '@edu.iot', HIRE_DATE, SYSDATE
FROM HR.EMPLOYEES;
```

모델 객체 정의

❖ Member 모델

```
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Member {
    private String     userId;
    private String     name;
    private String     password;
    private UserLevel  userLevel;
    private String     phoneNumber;
    private String     nickName;
    private String     email;
    private String     address;
    private Date       regDate;
    private Date       updateDate;
}
```

모델 객체 정의

❖ mybatis-config.xml

- ○ 모델 객체의 alias 설정

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
   :
   <typeAliases>
      <typeAlias  type="edu.iot.sagittarius.model.Member"
                  alias="Member"/>
   </typeAliases>
   :

   <!-- Mapper 파일 위치 설정 -->
   <mappers></mappers>

</configuration>
```

Mapper

❖ 인터페이스 준비

- CrudDao
- Pagination
- Random

Mapper

❖ MemberDao 인터페이스

```
public interface MemberDao extends CrudDao<Member, String>{

}
```

❖ Mapper

- src/java/resources 소스 폴더에 mapper 패키지 생성
  - member-mapper.xml 파일 생성

- mybatis-config.xml
  - mapper 파일의 경로 등록

```
    :
    <!-- Mapper 파일 위치 설정 -->
    <mappers>
       <mapper resource="mapper/member-mapper.xml"/>
    </mappers>

</configuration>
```

❖ Mapper

- Dao에서 실행할 SQL문을 XML 엘리먼트로 정의
- 기본 골격

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
   PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
   "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="edu.iot.sagittarius.dao.MemberDao">
</mapper>
```

  - namespace 속성
    - 식별자 구분을 위한 prefix(자바의 패키지 역할에 해당)
    - Dao 인터페이스의 전체 클래스명으로 지정

    namespace="edu.iot.sagittarius.dao.MemberDao"

❖ Mapper

- 인터페이스 구현

```
public class CrudDaoImpl<M, K> implements CrudDao{
   // Mapper의 namepsace와 동일하게 구성
   protected String namespace;

   public CrudDaoImpl(String name) {
      this.namespace =
            getClass().getPackage().getName() + "." + name;
   }
   :
}
```

❖ Mapper

- 인터페이스 구현 : sql 문 실행 골격
  - SqlSession 객체로 Mapper의 SQL 엘리먼트 실행
  - SqlSession 객체 얻기

```
try(SqlSession session=Session.getSession()) {
   // SqlSession 메서드로 대응하는 sql 쿼리 실행

}
```

Mapper

- SqlSession 객체 주요 메서드
  - List selectList(String)
  - List selectList(String, Object)
  - Object selectOne(String)
  - Object selectOne(String, Object)
  - int insert(String)
  - int insert(String, Object)
  - int update(String)
  - int update(String, Object)
  - int delete(String)
  - int delete(String, Object)
    - 첫 번째 인자 : 엘리먼트 식별문자열(namespace.엘리먼트 id)
    - 두 번째 인자 : 전달 값

❖ Mapper

- SQL문 엘리먼트
  - <select>select sql문</select>
  - <insert>insert sql문</insert>
  - <update>update sql문</update>
  - <delete>delete sql문</delete>

- 연관된 Dao 인터페이스의 각 메서드에 대응하는 엘리먼트를 정의
- Dao 인터페이스 구현 클래스에서 SqlSession 메서드로 실행
  - <select> → selectOne() 또는 selectList()로 실행
  - <insert> → insert()로 실행
  - <update> → update()로 실행
  - <delete> → delete()로 실행

❖ Mapper

- ㅇ 엘리먼트의 속성
  - ▪ id : 인터페이스 메서드 명에 해당
  - ▪ parameterType : 인터페이스 메서드의 매개변수 타입에 해당
    - 없거나 1개 지정
      - > 없는 경우 속성 생략
      - > 1개인 경우 : 프리미티브 타입, 모델 객체, Map 객체
  - ▪ resultType : 인터페이스 메서드의 리턴 타입에 해당
    - sql 문의 리턴값을 매칭
    - select 문
      - > selectList()로 호출된 경우 List<resultType>으로 리턴
      - > selectOne()으로 호출된 경우 resultType 단일 객체로 리턴
      - >
    - insert, update, delete 문
      - > int 타입으로 고정되어 있으므로 지정 생략

❖ select 문 처리

- 인터페이스

```
int count() throws Exception;
```

- 매퍼

```
<select id="count" resultType="int"><![CDATA[
   select count(*) from member
]]></select>
```

- <![CDATA[…]]>
  - <, > 등은 xml 태그 구분용으로 사용되는 문자
  - 데이터로서 사용하는 경우 <![CDATA[…]]> 안에 기술
    - xml 파서에게 분석된 내용이므로 그냥 데이터로 처리하도록 알림

❖ select 문 처리

- 인터페이스 구현 : count

```
public int count() throws Exception {
   try(SqlSession session=Session.getSession()) {
      return session.selectOne(namespace + ".count");
   }
}
```

❖ select 문 처리

- 인터페이스

```
List<Member> getList() throws Exception;;
```

- 매퍼

```
<select id="getList" resultType="Member"><![CDATA[
   select * from member
]]></select>
```

❖ select 문 처리

- 인터페이스 구현 : getList

```
public List<M> getList() throws Exception {
   try(SqlSession session=Session.getSession()) {
      return session.selectList(namespace + ".getList");
   }
}
```

❖ select 문 처리 (키에 대한 검색)

- 인터페이스

```
Member findById(String userId) throws Exception;;
```

- 매퍼

```
<select id="findById" parameterType="string" resultType="Member">
<![CDATA[
   select * from member where user_id = #{userId}
]]></select>
```

  - #{프로퍼티명}
    - PreparedStatement의 ? 처리와 동일
    - ? 위치에 parameterType 객체의 프로퍼티명 기술

❖ Mapper

- #{프로퍼티명} 해석, 예 : #{userId}
  - 인자가 프리미티브 또는 String 인 경우(단 일값)
    - 프로퍼티 명과 관계 없이 지정
  - 인자가 모델 객체인 경우
    - getter 호출
    - getUserId();
  - 인자가 Map인 경우
    - get() 호출
    - get("userId")

❖ select 문 처리

- 인터페이스 구현 : findById

```
public M findById(K k) throws Exception {
   try(SqlSession session=Session.getSession()) {
      return session.selectOne(namespace + ".findById", k);
   }
}
```

❖ insert 문 처리

- 인터페이스

```
int insert(Member member) throws Exception;
```

- 매퍼

```
<insert id="insert" parameterType="Member"><![CDATA[
   insert into member (
      user_id, name, password, email, user_level,
      phone_number, nick_name, address
   )
   values(
      #{userId}, #{name}, #{password} , #{email}, #{userLevel},
      #{phoneNumber}, #{nickName}, #{address}
   )
]]></insert>
```

❖ insert 문 처리

- 인터페이스 구현 : insert

```
public int insert() throws Exception {
   try(SqlSession session=Session.getSession()) {
      return session.insert(namespace + ".insert");
   }
}
```

❖ update 문 처리

- 인터페이스

```
int update(Member member) throws Exception;
```

- 매퍼

```
<update id="update" parameterType="Member"><![CDATA[
   update member set
      email=#{email},
      phone_number=#{phoneNumber},
      nick_name=#{nickname},
      address=#{address},
      update_date = sysdate
   where user_id=#{userId} and password=#{password}
]]></update>
```

❖ update 문 처리

- 인터페이스 구현 : update

```
public int update () throws Exception {
   try(SqlSession session=Session.getSession()) {
      return session.insert(namespace + ".update");
   }
}
```

❖ delete 문 처리

- 인터페이스

```
int delete(String userId) throws Exception;
```

- 매퍼

```
<delete id="delete" parameterType="String"><![CDATA[
   delete from member
   where user_id = #{userId}
]]>
</delete>
```

❖ delete 문 처리

- 인터페이스 구현 : delete

```
public int delete() throws Exception {
   try(SqlSession session=Session.getSession()) {
      return session.insert(namespace + ".delete");
   }
}
```

Dao 구현

❖ MemberDao 인터페이스

```
public interface MemberDao extends CrudDao<Member, String>{

}
```

Dao 구현

❖ MemberDaoImpl

```
public class MemberDaoImpl
                extends CrudDaoImpl<Member, String>
                implements MemberDao{

   public MemberDaoImpl() {
      super("MemberDao");
   }

}
```

Dao 구현

❖ MemberDaoEx1

```
public class MemberDaoEx1 {
   public static void main(String[] args) {
      MemberDao dao = new MemberDaoImpl();

      try {
         System.out.println(dao.count());
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

❖ PaginationDao 인터페이스

```
public interface PaginationDao<M> {
   List<M> getPage(int start, int end) throws Exception;
}
```

❖ member-mapper.xml

```
<select id="getPage"   parameterType="map"
                       resultType="Member"><![CDATA[
      select * from member_pagination_view
      where seq between #{start} and #{end}
   ]]></select>
```

❖ PaginationDaoImpl

```
public class PaginationDaoImpl<M, K> extends CrudDaoImpl<M, K> {

   public PaginationDaoImpl(String name) {
      super(name);
   }

   public List<M> getPage(int start, int end) throws Exception {
      Map<String, Integer> map = new HashMap<>();
      map.put("start", start);
      map.put("end", end);

      try(SqlSession session=Session.getSession()) {
         return session.selectOne(namespace + ".getPage", map);
      }
   }
}
```