MapStruct

소스 : https://github.com/hyomee/JAVABASIC

Content

1. pom.xml 설정
2. 동일 이름 매핑
3. Spring과 CDI (Contexts and Dependency Injection)
4. 다른 이름 매핑
5. 여러 객체를 하나로
6. 리스트 객체
7. 자식 매핑
8. 기존 인스턴스 객체 update
9. 타입 매핑, 기본값, NULL, 표현식
10. 매핑 전. 후 처리 및 Collection Mapping
11. 정책

1. 기본

MapStruct는 java Bean 간의 매핑을 구현을 단순화하는 코드 생성기
Interface를 생성 한 후 컴파일을 하면 자동으로 구현채를 생성 함.

01. pom..xml

```
- <dependency>
    <groupId>org.mapstruct</groupId>
    <artifactId>mapstruct</artifactId>
    <version>1.4.0.Final</version>
  </dependency>

- mapstruct-processor 구성

<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.5.1</version>
    <configuration>
        <source>1.8</source>
        <target>1.8</target>
        <annotationProcessorPaths>
            <path>
                <groupId>org.mapstruct</groupId>
                <artifactId>mapstruct-processor</artifactId>
                <version>1.3.1.Final</version>
            </path>
        </annotationProcessorPaths>
    </configuration>
</plugin>
```

```java
@Mapper
public interface SimpleSourceToTargetMapper {
  SimpleSourceToTargetMapper SIMPLE_SOURCE_TO_TARGET_MAPPER = Mappers.getMapper(SimpleSourceToTargetMapper.class);
  SimpleTargetDTO sourceDTOToTargetDTO(SimpleSourceDTO simpleSourceDTO);
  SimpleSourceDTO targetDTOToSoreceDTO(SimpleTargetDTO simpleTargetDTO);
}
```

mvn clean install (compile)

```java
@Generated(
    value = "org.mapstruct.ap.MappingProcessor")
@Component
public class SimpleSourceToTargetMapperImpl implements SimpleSourceToTargetMapper {

    @Override
    public SimpleTargetDTO sourceDTOToTargetDTO(SimpleSourceDTO simpleSourceDTO) {
        if ( simpleSourceDTO == null ) {
            return null;
        }
        SimpleTargetDTO simpleTargetDTO = new SimpleTargetDTO();
        simpleTargetDTO.setCustNo( simpleSourceDTO.getCustNo() );
        simpleTargetDTO.setCustNm( simpleSourceDTO.getCustNm() );
        return simpleTargetDTO;
    }

    @Override
    public SimpleSourceDTO targetDTOToSoreceDTO(SimpleTargetDTO simpleTargetDTO) {
        if ( simpleTargetDTO == null ) {
            return null;
        }
        SimpleSourceDTO simpleSourceDTO = new SimpleSourceDTO();
        simpleSourceDTO.setCustNo( simpleTargetDTO.getCustNo() );
        simpleSourceDTO.setCustNm( simpleTargetDTO.getCustNm() );

        return simpleSourceDTO;
    }
}
```

2. 동일 이름 매핑

01. 소스

```java
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class SimpleSourceDTO {

  private String custNo;
  private String custNm;

}
```

```java
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class SimpleTargetDTO {
  private String custNo;
  private String custNm;
}
```

```java
@Mapper
public interface SimpleSourceToTargetMapper {
  SimpleSourceToTargetMapper SIMPLE_SOURCE_TO_TARGET_MAPPER = Mappers.getMapper(SimpleSourceToTargetMapper.class);
  SimpleTargetDTO sourceDTOToTargetDTO(SimpleSourceDTO simpleSourceDTO);
}
```

매핑

mvn clean install (compile)

```java
@Generated(
    value = "org.mapstruct.ap.MappingProcessor",
    date = "2021-01-04T22:33:37+0900",
    comments = "version: 1.4.1.Final, compiler: javac, environment: Java 11.0.2 (Oracle Corporation)"
)
@Component
public class SimpleSourceToTargetMapperImpl implements SimpleSourceToTargetMapper {
    @Override
    public SimpleTargetDTO sourceDTOToTargetDTO(SimpleSourceDTO simpleSourceDTO) {
        if ( simpleSourceDTO == null ) {
            return null;
        }
        SimpleTargetDTO simpleTargetDTO = new SimpleTargetDTO();
        simpleTargetDTO.setCustNo( simpleSourceDTO.getCustNo() );
        simpleTargetDTO.setCustNm( simpleSourceDTO.getCustNm() );
        return simpleTargetDTO;
    }}
```

```java
@Test
void contextLoads() {
  SimpleSourceDTO simpleSourceDTO = SimpleSourceDTO.of("홍길동", "A0001");
  SimpleTargetDTO simpleTargetDTO = SimpleSourceToTargetMapper.SIMPLE_SOURCE_TO_TARGET_MAPPER.sourceDTOToTargetDTO(simpleSourceDTO);

  assertEquals(simpleSourceDTO.getCustNm(), simpleTargetDTO.getCustNm());
  assertEquals(simpleSourceDTO.getCustNo(), simpleTargetDTO.getCustNo());
}
```

3. Spring과 CDI

MapStruct 의존성 주입

01. 의존성 주입

- Spring 사용 :: @Mapper(componentModel = " spring ")
- Spring 미사용 :: @Mapper(componentModel = " cdi ")

```java
@Mapper(componentModel = "spring")
public interface SimpleToTargetInjection {
  SimpleTargetDTO sourceToTarget(SimpleSourceDTO simpleSourceDTO);
  SimpleSourceDTO targetToSorece(SimpleTargetDTO simpleTargetDTO);
}
```

```java
@Generated(
    value = "org.mapstruct.ap.MappingProcessor",
    date = "2021-01-04T23:21:52+0900",
    comments = "version: 1.4.1.Final, compiler: javac, environment: Java 11.0.2 (Oracle Corporation)"
)

@Component
public class SimpleToTargetInjectionImpl implements SimpleToTargetInjection {

    @Override
    public SimpleTargetDTO sourceToTarget(SimpleSourceDTO simpleSourceDTO) {
        if ( simpleSourceDTO == null ) {
            return null;
        }

        SimpleTargetDTO simpleTargetDTO = new SimpleTargetDTO();

        simpleTargetDTO.setCustNo( simpleSourceDTO.getCustNo() );
        simpleTargetDTO.setCustNm( simpleSourceDTO.getCustNm() );

        return simpleTargetDTO;
    }
.....

}
```

```java
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class SimpleSourceDTO {

  private String custNo;
  private String custNm;

}
```

```java
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class SimpleTargetDTO {
  private String custNo;
  private String custNm;
}
```

```java
@Autowired
private SimpleToTargetInjection simpleToTargetInjection ;

@Test
void contextLoads() {
  SimpleSourceDTO simpleSourceDTO = SimpleSourceDTO.of("홍길동", "A0001");
  SimpleTargetDTO simpleTargetDTO = simpleToTargetInjection.sourceToTarget(simpleSourceDTO);

  assertEquals(simpleSourceDTO.getCustNm(), simpleTargetDTO.getCustNm());
  assertEquals(simpleSourceDTO.getCustNo(), simpleTargetDTO.getCustNo());
}
```

ABACUS

4. 다른 동일 이름 매핑

01. @Mapping은 컬럼 단위 정의, 여러 개인 경우는 @Mappings를 사용을 하여서 작성

```
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class SimpleSourceDTO {

  private String custNo;
  private String custNm;

}
```

```
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class SimpleDifferentTargetDTO {
  private String custNumber;
  private String custNm;
  private String ages;
}
```

```
@Mapper(componentModel = "spring")
public interface DiffrentMapStruct {
  @Mappings({
        @Mapping(target="custNumber", source = "custNo")
  })
  SimpleDifferentTargetDTO simpleToDifferent(SimpleSourceDTO simpleSourceDTO);
}
```

매핑

mvn clean install (compile)

```
@Generated(
    value = "org.mapstruct.ap.MappingProcessor"
)
@Component
public class DiffrentMapStructImpl implements DiffrentMapStruct {

    @Override
    public SimpleDifferentTargetDTO simpleToDifferent(SimpleSourceDTO simpleSourceDTO) {
        if ( simpleSourceDTO == null ) {
            return null;
        }
        SimpleDifferentTargetDTO simpleDifferentTargetDTO = new SimpleDifferentTargetDTO();

        simpleDifferentTargetDTO.setCustNumber( simpleSourceDTO.getCustNo() );
        simpleDifferentTargetDTO.setCustNm( simpleSourceDTO.getCustNm() );

        return simpleDifferentTargetDTO;
    }
}
```

```
@Autowired
private DiffrentMapStruct diffrentMapStruct ;

@Test
void contextLoads() {
  SimpleSourceDTO simpleSourceDTO = SimpleSourceDTO.of("홍길동", "A0001");
  SimpleDifferentTargetDTO simpleDifferentTargetDTO = diffrentMapStruct.simpleToDifferent(simpleSourceDTO);

  assertEquals(simpleSourceDTO.getCustNm(), simpleDifferentTargetDTO.getCustNm());
  assertEquals(simpleSourceDTO.getCustNo(), simpleDifferentTargetDTO.getCustNumber());
}
```

5. 여러 객체를 하나로

01. @Mapping을 사용해서 각각 매핑

```
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class UserDTO {
  private String userNo;
  private String userNm;
  private String ages;
}
```

```
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class AddressDTO {
  private String userNo;
  private String postNo;
  private String address;
}
```

```
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class UserAddressDTO {
  private String userNo;
  private String userNm;
  private String nai;
  private String address;
}
```

```
@Mapper(componentModel = "spring")
public interface Composer {

  @Mapping(target = "nai", source = "userDTO.ages")
  @Mapping(target = "userNo", source = "userDTO.userNo")
  UserAddressDTO toUserAddress(UserDTO userDTO, AddressDTO
addressDTO);

}
```

** 같은 이름을 재외 한 다른 이름의 필드를 @Mapping을 사용해서 정의
source는 객체.필드를 사용 하여 정확히 표시 함
=> 표시 하지 않으면 빌드 오류 발생

```
@Generated(
    value = "org.mapstruct.ap.MappingProcessor"
)
@Component
public class ComposerImpl implements Composer {

    @Override
    public UserAddressDTO toUserAddress(UserDTO userDTO, AddressDTO addressDTO) {
        if ( userDTO == null && addressDTO == null ) {
            return null;
        }

        UserAddressDTO userAddressDTO = new UserAddressDTO();

        if ( userDTO != null ) {
            userAddressDTO.setNai( userDTO.getAges() );
            userAddressDTO.setUserNo( userDTO.getUserNo() );
            userAddressDTO.setUserNm( userDTO.getUserNm() );
        }
        if ( addressDTO != null ) {
            userAddressDTO.setAddress( addressDTO.getAddress() );
        }

        return userAddressDTO;
    }
}
```

6. 리스트 객체

01. @Mapping을 사용해서 각각 매핑

```java
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class UserAddDTO {
  private String userNo;
  private String userNm;
  private String ages;
  private List<AddressDTO> addressDTOList;
  private List<MemberDTO> memberDTOList;
}
```

```java
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class AddressDTO {
  private String userNo;
  private String postNo;
  private String address;
}
```

```java
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class MemberDTO {
  private String userNo;
  private String memberNo;
  private String memberNm;
  private String ages;
}
```

```java
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class TargetUserAddDTO {
  private String userNo;
  private String userNm;
  private String ages;
  private List<AddressDTO> addressDTOS;
  private List<MemberDTO> memberDTOList;
}
```

동일이름 자동 매핑

```java
@Mapper(componentModel = "spring")
public interface usertToTarget {

  @Mapping(target="addressDTOS", source = "addressDTOList")
  TargetUserAddDTO toDto(UserAddDTO userAddDTO);
}
```

```java
@Generated(
    value = "org.mapstruct.ap.MappingProcessor")
@Component
public class usertToTargetImpl implements usertToTarget {

    @Override
    public TargetUserAddDTO toDto(UserAddDTO userAddDTO) {
        if ( userAddDTO == null ) {
            return null;
        }

        TargetUserAddDTO targetUserAddDTO = new TargetUserAddDTO();

        List<AddressDTO> list = userAddDTO.getAddressDTOList();
        if ( list != null ) {
            targetUserAddDTO.setAddressDTOS( new ArrayList<AddressDTO>( list ) );
        }

        targetUserAddDTO.setUserNo( userAddDTO.getUserNo() );
        targetUserAddDTO.setUserNm( userAddDTO.getUserNm() );
        targetUserAddDTO.setAges( userAddDTO.getAges() );

        List<MemberDTO> list1 = userAddDTO.getMemberDTOList();
        if ( list1 != null ) {
            targetUserAddDTO.setMemberDTOList( new ArrayList<MemberDTO>( list1 ) );
        }

        return targetUserAddDTO;
    }
}
```

다른 이름

7. 자식 매핑

01. @Mapping, uses 을 사용해서 각각 매핑

```
public class UserParent {
  private String userNo;
  private String userNm;
  private List<AddressChild> addressChildList;
}
```

```
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class AddressChild {
  private String userNo;
  private String adderssNo;
  private String address;
}
```

```
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class UserParentDTO {
  private String userNo;
  private String userNm;
  private List<AddressChildDTO>
addressChildDTOList;
}
```

```
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class AddressChildDTO {
  private String userNo;
  private String adderssNo;
  private String address;
}
```

```
@Mapper(componentModel = "spring", uses = {AddressChildMapper.class})
public interface UserParentMapper {

  @Mapping(target = "addressChildDTOList", source = "userParent.addressChildList")

  UserParentDTO toDto(UserParent userParent);
}
```

```
@Generated(
    value = "org.mapstruct.ap.MappingProcessor")
@Component
public class UserParentMapperImpl implements UserParentMapper {

    @Autowired
    private AddressChildMapper addressChildMapper;

    @Override
    public UserParentDTO toDto(UserParent userParent) {
        if ( userParent == null ) {
            return null;
        }

        UserParentDTO userParentDTO = new UserParentDTO();

        userParentDTO.setAddressChildDTOList( addressChildListToAddressChildDTOList( userParent.getAddressChildList() ) );
        userParentDTO.setUserNo( userParent.getUserNo() );
        userParentDTO.setUserNm( userParent.getUserNm() );

        return userParentDTO;
    }

    protected List<AddressChildDTO> addressChildListToAddressChildDTOList(List<AddressChild> list) {
        if ( list == null ) {
            return null;
        }

        List<AddressChildDTO> list1 = new ArrayList<AddressChildDTO>( list.size() );
        for ( AddressChild addressChild : list ) {
            list1.add( addressChildMapper.toDto( addressChild ) );
        }

        return list1;
    }
}
```

8. 기존 인스턴스 update

01. @MappingTarget를 사용 하여 정의

```java
public class UserParent {
  private String userNo;
  private String userNm;
  private List<AddressChild> addressChildList;
}
```

```java
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class AddressChild {
  private String userNo;
  private String adderssNo;
  private String address;
}
```

```java
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class UserParentDTO {
  private String userNo;
  private String userNm;
  private List<AddressChildDTO>
addressChildDTOList;
}
```

```java
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class AddressChildDTO {
  private String userNo;
  private String adderssNo;
  private String address;
}
```

```java
@Mapper(componentModel = "spring", uses = {AddressChildMapper.class})
public interface UserParentMapper {

  @Mapping(target = "addressChildDTOList", source = "userParent.addressChildList")

  void updateUserParentDto(UserParent userParent,
                           @MappingTarget UserParentDTO userParentDTO);
}
```

```java
@Generated(
    value = "org.mapstruct.ap.MappingProcessor")
@Component
public class UserParentMapperImpl implements UserParentMapper {

    @Autowired
    private AddressChildMapper addressChildMapper;

@Override
public void updateUserParentDto(UserParent userParent, UserParentDTO userParentDTO) {
    if ( userParent == null ) {
        return;
    }

    if ( userParentDTO.getAddressChildDTOList() != null ) {
        List<AddressChildDTO> list = addressChildListToAddressChildDTOList( userParent.getAddressChildList() );
        if ( list != null ) {
            userParentDTO.getAddressChildDTOList().clear();
            userParentDTO.getAddressChildDTOList().addAll( list );
        }
        else {
            userParentDTO.setAddressChildDTOList( null );
        }
    }
    else {
        List<AddressChildDTO> list = addressChildListToAddressChildDTOList( userParent.getAddressChildList() );
        if ( list != null ) {
            userParentDTO.setAddressChildDTOList( list );
        }
    }
    userParentDTO.setUserNo( userParent.getUserNo() );
    userParentDTO.setUserNm( userParent.getUserNm() );
}
```

9. 타입 매핑, 기본값, NULL, 표현식

01. 타입 매핑

```java
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class UserType {
  private long id;
  private String userId;
  private String userNm;
  private Date birthday;
  private String pay;
  private LocalDateTime currentDate;
  private String desc;
}
```

```java
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(staticName = "of")
public class UserTypeDTO {
  private long id;
  private String userId  ;  // defaultExpression
  private String userNm;
  private String birthday; // dateFormat
  private long pay;        // numberFormat
  private LocalDateTime currentDate;
  private String desc;
}
```

```java
@Mapper(componentModel = "string",
        // ERROR, IGNORE, WARN 정책
        unmappedTargetPolicy = ReportingPolicy.ERROR,
        // Source가 null이거나 혹은 Source의 특정 필드가 null인 경우
        nullValueMappingStrategy = NullValueMappingStrategy.RETURN_NULL,
        imports = {LocalDateTime.class, UUID.class})
public interface DataTypeMap {

  @Mapping(target="id", constant = "-1L")

  @Mapping(target="userId", expression = "java(UUID.randomUUID().toString())")

  @Mapping(target="userNm", source = "userNm", defaultValue = "이름")

  @Mapping(target="birthday", source = "birthday", dateFormat = "dd-MM-yyyy HH:mm:ss")

  @Mapping(target="pay", source = "pay", numberFormat = "$#.00")

  @Mapping(target="currentDate", source = "currentDate",
                  defaultExpression = "java(LocalDateTime.now())")

  // 특정 필드는 매핑되지 않길 원한다면 @Mapping 어노테이션에 ignore = true 속성
   @Mapping(target="desc", source = "desc", ignore = true)
  UserTypeDTO toDto(UserType userType);
}
```

```java
@Generated(
    value = "org.mapstruct.ap.MappingProcessor"
)
public class DataTypeMapImpl implements DataTypeMap {

    @Override
    public UserTypeDTO toDto(UserType userType) {
        if ( userType == null ) {
            return null;
        }

        UserTypeDTO userTypeDTO = new UserTypeDTO();

        if ( userType.getUserNm() != null ) {
            userTypeDTO.setUserNm( userType.getUserNm() );
        }
        else {
            userTypeDTO.setUserNm( "이름" );
        }
        if ( userType.getBirthday() != null ) {
            userTypeDTO.setBirthday( new SimpleDateFormat( "dd-MM-yyyy HH:mm:ss" ).format( userType.getBirthday() ) );
        }
        try {
            if ( userType.getPay() != null ) {
                userTypeDTO.setPay( new DecimalFormat( "$#.00" ).parse( userType.getPay() ).longValue() );
            }
        }
        catch ( ParseException e ) {
            throw new RuntimeException( e );
        }
        if ( userType.getCurrentDate() != null ) {
            userTypeDTO.setCurrentDate( userType.getCurrentDate() );
        }
        else {
            userTypeDTO.setCurrentDate( LocalDateTime.now() );
        }

        userTypeDTO.setId( -1L );
        userTypeDTO.setUserId( UUID.randomUUID().toString() );

        return userTypeDTO;
    }
}
```

10. 매핑 전. 후 처리 및 Collection Mapping

01. @BeforeMapping 및 @AfterMapping

```
@Mapper(uses = {PatientMapper.class}, componentModel = "spring")
public abstract class DoctorCustomMapper {

  @BeforeMapping
  protected void validate(Doctor doctor) {
    if(doctor.getPatientList() == null){
      doctor.setPatientList(new ArrayList<>());
    }
  }

  @AfterMapping
  protected void updateResult(@MappingTarget DoctorDto doctorDto) {
    doctorDto.setName(doctorDto.getName().toUpperCase());
    doctorDto.setDegree(doctorDto.getDegree().toUpperCase());
    doctorDto.setSpecialization(doctorDto.getSpecialization().toUpperCase());
  }

  @Mapping(source = "doctor.patientList", target = "patientDtoList")
  @Mapping(source = "doctor.specialty", target = "specialization")
  public abstract DoctorDto toDoctorDto(Doctor doctor);
}
```

02. Collection Mapping

```
@Mapper
public interface EmployeeMapper {
  List<EmployeeDTO> map(List<Employee> employees);
}

@Mapper
public interface EmployeeMapper {
  Set<EmployeeDTO> map(Set<Employee> employees);
}
```

https://www.baeldung.com/java-mapstruct-mapping-collections

11. 정책

정책	값	설명
unmappedSourcePolicy	IGNORE(default), WARN, ERROR	Source의 필드가 Target에 매핑되지 않을 때 정책이다. 예, ERROR로 설정하면 매핑 시 Source.aField가 사용되지 않는다면 컴파일 오류가 발생시킨다.
unmappedTargetPolicy	IGNORE, WARN(default), ERROR	Target의 필드가 매핑되지 않을 때 정책이다. 예, ERROR로 설정하면 매핑 시 Target.aField에 값이 매핑되지 않는다면 컴파일 오류가 발생시킨다.
typeConversionPolicy	IGNORE(default), WARN, ERROR	타입 변환 시 유실이 발생할 수 있을 때 정책이다. 예, ERROR로 설정하면 long에서 int로 값을 넘길 때 값에 유실이 발생할 수 있다. 이런 경우에 컴파일 오류를 발생시킨다.

전략	값	설명
nullValueMappingStrategy	RETURN_NULL(default), RETURN_DEFAULT	Source가 null일 때 정책이다.
nullValuePropertyMappingStrategy	SET_TO_NULL(default), SET_TO_DEFAULT, IGNORE	Source의 필드가 null일 때 정책이다.

THANKS

ABACUS

www.iabacus.co.kr

Tel. 82-2-2109-5400

Fax. 82-2-6442-5409