1. MapStruct (efficient Dto converter)

1.1. Configuration maven pour MapStruct

pom.xml

```
...
<properties>
        <mapstruct.version>1.5.2.Final</mapstruct.version>
        <m2e.apt.activation>jdt_apt</m2e.apt.activation>
</properties>
<dependencies>
                ...
                <dependency>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                        <optional>true</optional>
                </dependency>

                <dependency>
                        <groupId>org.mapstruct</groupId>
                        <artifactId>mapstruct</artifactId>
                        <version>${mapstruct.version}</version>
                </dependency>
                ...
        </dependency>
</dependencies>
<build>
<plugins>
        <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.10.1</version>
                <configuration>
                        <source>${java.version}</source>
                        <target>${java.version}</target>
                        <release>${java.version}</release>
                        <annotationProcessorPaths>
                                <path>
                                        <groupId>org.projectlombok</groupId>
                                        <artifactId>lombok</artifactId>
                                        <version>1.18.24</version>
                                </path>
                                <!-- Mapstruct should follow the lombok path(s) -->
                                <path>
                                        <groupId>org.mapstruct</groupId>
                                        <artifactId>mapstruct-processor</artifactId>
                                        <version>${mapstruct.version}</version>
                                </path>
                        </annotationProcessorPaths>
                </configuration>
        </plugin>      ...
</plugins></build></project>
```

1.2. Intégration de MapStruct dans l'IDE eclipse

pour que mapStruct s'intégre bien à eclipse:

1) installer le plugin ***m2e-apt*** via **Help / Eclipse marketPlace**

2) **project/properties .../**
   maven /annotation processing
     /enable specific
        / automatically configure JDT APT

1.3. Utilisation de MapStruct

MyMapper.java

```
package tp.appliSpring.converter;

import org.mapstruct.InheritConfiguration;
import org.mapstruct.Mapper;
import org.mapstruct.Mapping;
import org.mapstruct.factory.Mappers;
import ....

//@Mapper // MyMapper.INSTANCE...
@Mapper(componentModel = "spring") //for @Autowired
public interface MyMapper {

        MyMapper INSTANCE = Mappers.getMapper( MyMapper.class );

        OperationDto operationToOperationDto(Operation source);
        Operation operationDtoOperation(OperationDto source);

        @Mapping(target="number", source="numero")
        @Mapping(target="firstName", source="prenom")
        @Mapping(target="lastName", source="nom")
        @Mapping(target="address", source="adresse")
        ClientDto clientToClientDto(Client source);

        @InheritConfiguration
        @Mapping(target="comptes", source="comptes" , resultType = CompteDto.class)
        ClientDtoEx clientToClientDtoEx(Client source);

        @Mapping(target="numero", source="number")
        @Mapping(target="prenom", source="firstName")
        @Mapping(target="nom", source="lastName")
        @Mapping(target="adresse", source="address")
        @Mapping(ignore = true, target="comptes" )
        Client clientDtoToClient(ClientDto source);

        @InheritConfiguration
        @Mapping(ignore = false, target="comptes" )
        Client clientDtoExToClient(ClientDtoEx source);

        CompteDto compteToCompteDto(Compte compte);
        Compte compteDtoToCompte(CompteDto compteDto);
}
```

La classe d'implémentation est automatiquement générée à partir de l'interface lors du build (maven et/ou eclipse)

Exemple de code généré (dans *\target\generated-sources\annotations\tp\appliSpring\converter\MyMapperImpl.java*) :

```
public class MyMapperImpl implements MyMapper {
..
public ClientDto clientToClientDto(Client source) {
      if ( source == null ) {
         return null;
      }
      ClientDto clientDto = new ClientDto();
      clientDto.setNumber( source.getNumero() );
      clientDto.setFirstName( source.getPrenom() );
      clientDto.setLastName( source.getNom() );
      clientDto.setAddress( source.getAdresse() );
      clientDto.setEmail( source.getEmail() );
      return clientDto;
   } ...
```

Utilisation sans injection Spring :

```
return MyMapper.INSTANCE.clientToClientDtoEx(client);
```

Utilisation avec injection Spring :

```
...
@Autowired //ou @Resource ou autre
private MyMapper myMapper;


public ClientDtoEx searchCustomerWithAccountsById(Long numClient) {
                Client client  = daoClient.findWithAccountById(numClient);
                return myMapper.clientToClientDtoEx(client);
        }
```

1.4. MyGenericMapper s'appuyant sur MapStruct

***MyGenericMapper*.java**

```
package tp.appliSpring.converter;

import java.lang.reflect.Method;  import java.util.List;  import java.util.stream.Collectors;
import org.springframework.beans.BeanUtils;

public class MyGenericMapper {

static MyMapper mapper = MyMapper.INSTANCE;

static String withFirstLowerCase(String s){
        return Character.toLowerCase(s.charAt(0)) + s.substring(1);
}

@SuppressWarnings("unchecked")
public static <S,D> D map(S source,Class<D> destinationClass) {
D destination = null;
try {
        //With mapStruct
        String convertMethodName=withFirstLowerCase(source.getClass().getSimpleName()
+ "To" + destinationClass.getSimpleName());
 //System.out.println("convertMethodName="+convertMethodName);
 Method convertMethod=mapper.getClass().getDeclaredMethod(convertMethodName,
 source.getClass());
 if(convertMethod!=null) {
 destination = (D) convertMethod.invoke(mapper, source);
 }else {
 //without mapStruct (as fault back)
 destination = destinationClass.getDeclaredConstructor().newInstance();
 BeanUtils.copyProperties(source, destination);
 }
} catch (Exception e) {
 e.printStackTrace();
}
return destination;
}

public static <S,D> List<D> map(List<S> sourceList , Class<D> destinationClass){
 return  sourceList.stream()
 .map((source)->map(source,destinationClass))
 .collect(Collectors.toList());
}
}
```

Exemple d'utilisation :

```
return MyGenericMapper.map(client,ClientDtoEx.class);
```