

# Mapstruct

## Step 1: Add dependency & Maven Plugin

```
pom.xml (demomaven)
39 <artifactId>postgresql</artifactId>
40 <scope>runtime</scope>
41 </dependency>
42
43 <dependency>
44 <groupId>org.springframework.boot</groupId>
45 <artifactId>spring-boot-starter-test</artifactId>
46 <scope>test</scope>
47 </dependency>
48
49 <dependency>
50 <groupId>org.mapstruct</groupId>
51 <artifactId>mapstruct</artifactId>
52 <version>${org.mapstruct.version}</version>
53 </dependency>
54 <dependency>
55 <groupId>org.springframework.boot</groupId>
56 <artifactId>spring-boot-starter-validation</artifactId>
57 </dependency>
58
```

```
<dependency>
<groupId>org.mapstruct</groupId>
<artifactId>mapstruct</artifactId>
<version>${org.mapstruct.version}</version>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-validation</artifactId>
</dependency>
</dependencies>

<build>
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>
<version>3.8.1</version>
<configuration>
<source>11</source>
<target>11</target>
<annotationProcessorPaths>
<path>
<groupId>org.mapstruct</groupId>
<artifactId>mapstruct-processor</artifactId>
<version>${org.mapstruct.version}</version>
</path>
<path>
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<version>${lombok.version}</version>
</path>
<path>
<groupId>org.projectlombok</groupId>
<artifactId>lombok-mapstruct-binding</artifactId>
<version>0.1.0</version>
</path>
</annotationProcessorPaths>
</configuration>
</plugin>
</plugins>
</build>
```

## Step 2: Define Department mapper to map Entity to Dto:

### 1. Using INSTANCE:

```
6 usages 1 implementation
@Mapper(unmappedTargetPolicy = ReportingPolicy.IGNORE)
public interface DepartmentMapper {

    2 usages
    DepartmentMapper INSTANCE = Mappers.getMapper(DepartmentMapper.class);

    2 usages 1 implementation
    DepartmentDto toDto(Department department);

    1 usage 1 implementation
    List<DepartmentDto> toDtos(List<Department> departments);

}
```

➔ Then using these Mapper class:

```
import java.util.List;

@Service
@RequiredArgsConstructor
public class DepartmentServiceImpl implements DepartmentService {

    4 usages
    private final DepartmentRepository departmentRepository;

    1 usage
    @Override
    public List<DepartmentDto> getAll() {
        return DepartmentMapper.INSTANCE.toDtos(departmentRepository.findAll());
    }

    1 usage
    @Override
    public List<DepartmentStatisticsDto> getDepartmentStatistics() {
        return departmentRepository.countEmployeesInDepartments();
    }

    @Override
    public List<DepartmentStatisticsDto> getDepartmentStatisticsByJPAQL() {
        return departmentRepository.countEmployeesInDepartmentsJPAQL();
    }

    1 usage
    @Override
    public DepartmentDto findById(Long departmentId) {
        Department department = departmentRepository.findById(departmentId).orElseThrow(DemoException::DepartmentNotFound);
        return DepartmentMapper.INSTANCE.toDto(department);
    }
}
```

## Using componentModel

```
import com.example.demomaven.entity.Employee;
import com.example.demomaven.service.dto.EmployeeDto;
import org.mapstruct.Mapper;
import org.mapstruct.Mapping;

import java.util.List;

3 usages 1 implementation
@Mapper(componentModel = "spring")
public interface EmployeeMapper {

    1 usage 1 implementation
    @Mapping(target = "departmentName", source = "department.name")
    EmployeeDto toDto(Employee employee);

    1 usage 1 implementation
    List<EmployeeDto> toDtos(List<Employee> employees);
}
```

➔ Then Using Employee mapper( using requiredArgsConstructor or using @Autowired)

```
import java.util.List;

@Service
@RequiredArgsConstructor
@Slf4j
public class EmployeeServiceImpl implements EmployeeService {

    1 usage
    private final EmployeeRepository employeeRepository;

    1 usage
    private final EmployeeMapper employeeMapper;

    1 usage
    @Override
    public List<EmployeeDto> getEmployeeByFirstName(String firstName) {

        return employeeMapper.toDtos(employeeRepository.getEmployeebyFirstname(firstName));
    }
}
```