JPA mapper and partial update

https://www.baeldung.com/spring-data-partial-update#:~:text=In%20fact%2C%20JPA%20defines%20two,the%20burden%20on%20the%20ORM

The annotation `@Mapper(componentModel = "spring")` is used in MapStruct, which is a code generation library for Java.

- **MapStruct** simplifies the implementation of mappings between Java bean types by generating mapping code at compile time. It is particularly useful in projects where you need to perform object-to-object mapping and want to avoid writing boilerplate mapping code manually.

- **`componentModel = "spring"`** is an attribute of the `@Mapper` annotation that tells MapStruct to generate mappers as Spring-managed beans, leveraging Spring's dependency injection capabilities. By setting `componentModel` to `"spring"`, MapStruct will generate mappers that can be easily injected into other Spring-managed beans, such as services or controllers.

When you annotate a mapper interface with `@Mapper(componentModel = "spring")`, it allows you to inject and use the mapper as a Spring bean, which can be autowired into other components of your Spring application.

For example:

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

    MyDto toDto(MyEntity entity);

    MyEntity toEntity(MyDto dto);
}
```

With `componentModel = "spring"`, you can inject and use `MyMapper` as a Spring bean in your application:

```java
@Service
public class MyService {

    @Autowired
    private MyMapper myMapper;
    public MyDto getDto() {
        MyEntity entity = new MyEntity();
        // populate entity
        return myMapper.toDto(entity);
    }

    public MyEntity getEntity() {
        MyDto dto = new MyDto();
        // populate dto
        return myMapper.toEntity(dto);
    }
}
```

This allows you to use the mapper in your Spring application seamlessly, without any additional configuration for dependency injection.