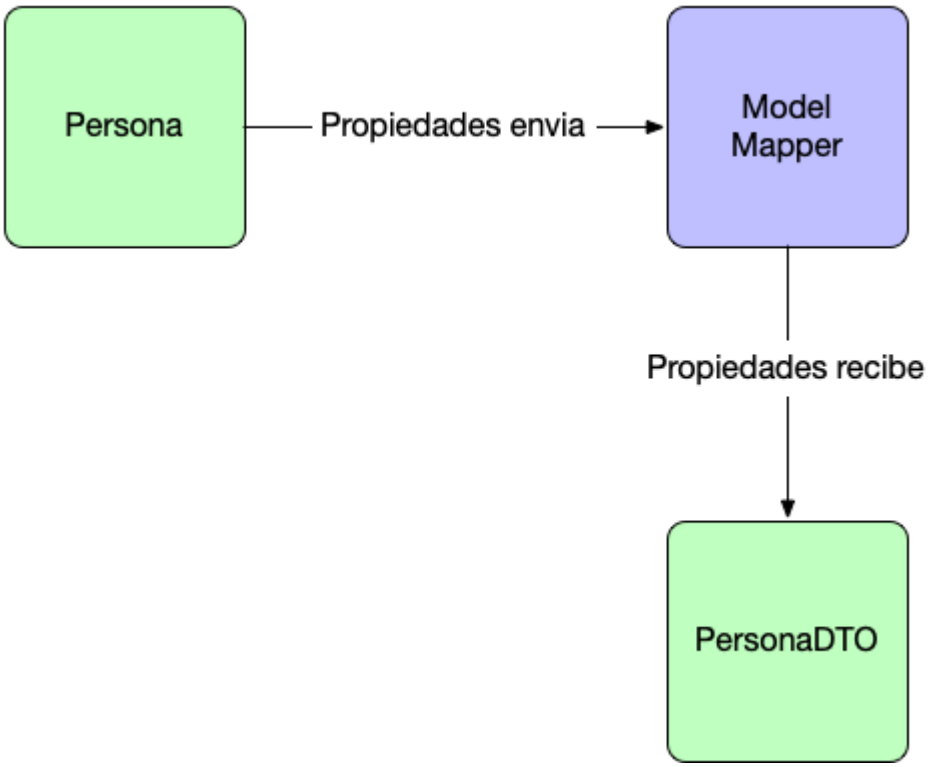
Vamos a hablar de Spring Copy Properties . La copia de propiedades y DTOs (Data Transfer Objects) es una necesidad común en Spring Framework cuando se trata de transferir datos entre objetos que comparten una estructura similar. Esto es especialmente relevante en la manipulación de objetos de negocio, como por ejemplo, entre el objeto 'Persona' y los DTOs, dado que sus campos a menudo son idénticos.



Spring Copy Properties y Model Mapper

En este artículo, exploraremos cómo abordar esta necesidad de manera eficiente utilizando Spring Boot. En primer lugar, vamos a importar la librería 'ModelMapper', que simplifica el proceso de mapeo de datos."

```
<dependency>
    <groupId>org.modelmapper</groupId>
    <artifactId>modelmapper</artifactId>
</dependency>
```

Una vez hecho esto construimos las dos clases que necesitamos para transferir unas propiedades de una a otra.

```
public class Persona {
    private String nombre;
    private int edad;

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }
}
```

```
public class PersonaDTO {
    private String nombre;
    private int edad;

    public String getNombre() {
        return nombre;
    }
```

```
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }
}
```

Nos creamos una clase de Servicio que se encargue de invocar al modelMapper y mapear las propiedades.

```
import org.modelmapper.ModelMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class PersonaService {

    @Autowired
    private ModelMapper modelMapper;

    public PersonaDTO copiarPropiedades(Persona persona) {
        return modelMapper.map(persona, PersonaDTO.class);
    }
}
```

Solo nos queda ver el código propio de Spring Boot que usa la anotación @Bean para iniciar

el mapeador

```
import org.modelmapper.ModelMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class SpringBootDTO {

    @Autowired
    private PersonaService personaService; // Inyecta el servicio

    @Bean
    public ModelMapper modelMapper() {
        return new ModelMapper();
    }

    public static void main(String[] args) {
        SpringApplication.run(SpringBootDTO.class, args);

        // Ahora puedes usar el servicio en el método main
        Persona persona = new Persona();
        persona.setNombre("Juan");
        persona.setEdad(30);

        PersonaDTO personaDTO =
personaService.copiarPropiedades(persona);

        System.out.println("Nombre (DTO): " + personaDTO.getNombre());
```

```
        System.out.println("Edad (DTO): " + personaDTO.getEdad());
    }
}
```

De esta forma hemos conseguido usar un ModelMapper y realizar un Spring Copy Properties de forma muy natural y sencila

Otros artículos relacionados

- ¿Que es GraalVM ?
- TypeScript interface utilizando Angular DTOs
- Utilizando Java Singleton Properties
- Entity to DTO y Java 8
- Java Properties Files y como usarlos
- ModelMapper