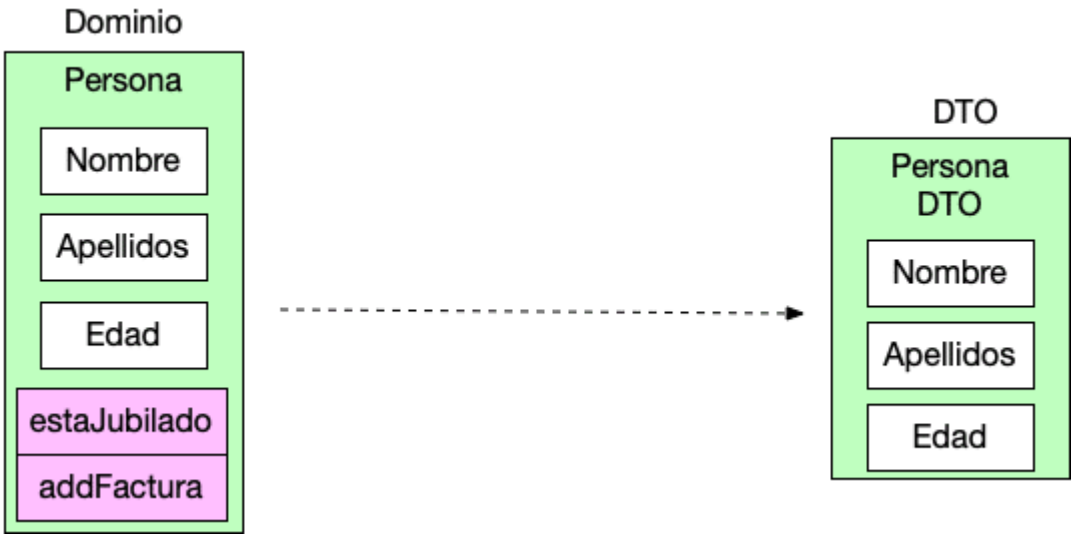
Tabla de Contenidos

- Transformaciones Sencillas
- El concepto de DTO Assembler
  - Data Transfer Objects y Agregados
  - DTO Assembler y el principio DRY
- Otros artículos relacionados

El concepto de DTO Assembler es un concepto relativamente sencillo . Los Data Transfer Objects o DTOs se usan para pasar información entre distintas capas y generar un mayor aislamiento entre los objetos de negocio y aplicaciones externas.

Transformaciones Sencillas

En muchos casos la transformación es muy directa y pasamos de una Persona como objeto de negocio a una PersonaDTO que contiene la misma información pero que es un DTO y no tiene métodos de negocio.



Veamos un ejemplo básico con Persona y PersonaDTO.

```
package com.arquitecturajava;
```

```
public class Persona {

	private String nombre;
	private String apellidos;
	private int edad;
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public String getApellidos() {
		return apellidos;
	}
	public void setApellidos(String apellidos) {
		this.apellidos = apellidos;
	}
	public int getEdad() {
		return edad;
	}
	public void setEdad(int edad) {
		this.edad = edad;
	}
	public Persona(String nombre, String apellidos, int edad) {
		super();
		this.nombre = nombre;
		this.apellidos = apellidos;
		this.edad = edad;
	}
	public boolean estaJubilado() {
		return edad>65;
```

```
        }
}
```

```
package com.arquitecturajava;

public class PersonaDTO {

        private String nombre;
        private String apellidos;
        private int edad;
        public String getNombre() {
                return nombre;
        }
        public void setNombre(String nombre) {
                this.nombre = nombre;
        }
        public String getApellidos() {
                return apellidos;
        }
        public void setApellidos(String apellidos) {
                this.apellidos = apellidos;
        }
        public int getEdad() {
                return edad;
        }
        public void setEdad(int edad) {
                this.edad = edad;
        }
        public PersonaDTO(Persona persona) {
                super();
                this.nombre = persona.getNombre();
                this.apellidos = persona.getApellidos();
```

```
                this.edad = persona.getEdad();
        }
}
```

Si queremos transformar la Persona en una PersonaDTO la transformación es muy muy sencilla. Bastaría con usar el constructor de PersonaDTO.

```
package com.arquitecturajava;

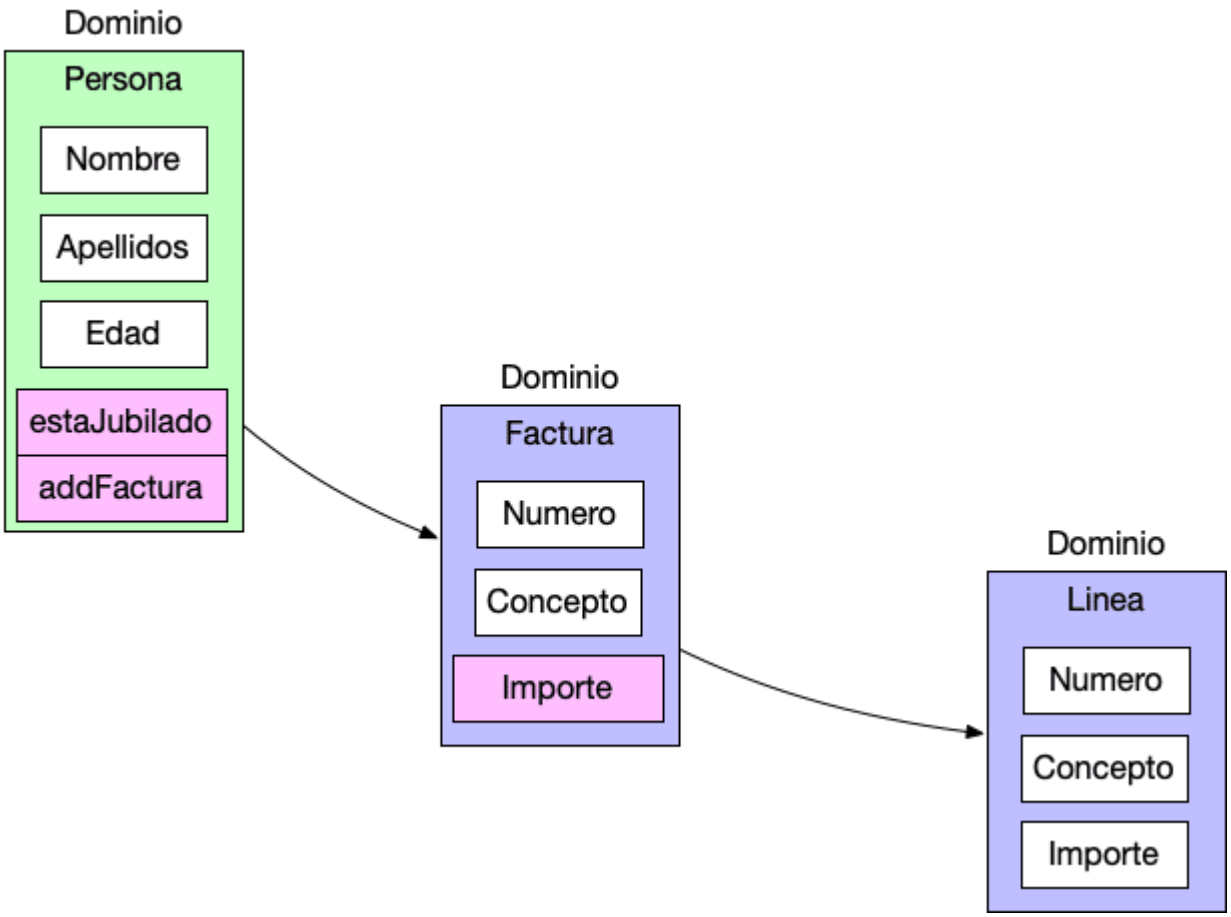
public class Principal {

        public static void main(String[] args) {
                Persona p= new Persona("juan","perez",20);
                PersonaDTO dto= new PersonaDTO(p);
        }

}
```

El concepto de DTO Assembler

Ahora bien existen situaciones en las cuales no tenemos algo tan sencillo sino que la estructura de la clase Persona es un grafo. Por ejemplo la Persona puede tener una serie de Facturas y estas una serie de Lineas.




```
package com.arquitecturajava.ejemplo2;

import java.util.ArrayList;
import java.util.List;

public class Persona {

	private String nombre;
	private String apellidos;
	private int edad;
	private List<Factura> facturas= new ArrayList<Factura>();
	public void addFactura(Factura f) {
		facturas.add(f);
```

```
}
public String getNombre() {
        return nombre;
}
public void setNombre(String nombre) {
        this.nombre = nombre;
}
public String getApellidos() {
        return apellidos;
}
public void setApellidos(String apellidos) {
        this.apellidos = apellidos;
}
public int getEdad() {
        return edad;
}
public void setEdad(int edad) {
        this.edad = edad;
}
public Persona(String nombre, String apellidos, int edad) {
        super();
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.edad = edad;
}
public boolean estaJubilado() {
        return edad>65;
}
public List<Factura> getFacturas() {
        return facturas;
}
```

```
        public void setFacturas(List<Factura> facturas) {
                this.facturas = facturas;
        }
}
```

```
package com.arquitecturajava.ejemplo2;

import java.util.ArrayList;
import java.util.List;

public class Factura {

        private int numero;
        private String concepto;
        private List<Linea> lineas = new ArrayList<Linea>();

        public int getNumero() {
                return numero;
        }

        public void addLinea(Linea linea) {
                lineas.add(linea);
        }

        public void setNumero(int numero) {
                this.numero = numero;
        }

        public String getConcepto() {
                return concepto;
        }
```

```
	public void setConcepto(String concepto) {
		this.concepto = concepto;
	}

	public List<Linea> getLineas() {
		return lineas;
	}

	public void setLineas(List<Linea> lineas) {
		this.lineas = lineas;
	}

	public Factura(int numero, String concepto) {
		super();
		this.numero = numero;
		this.concepto = concepto;
	}
	public int getImporteTotal() {
		int total=0;
		for (Linea l :lineas) {
			total+=l.getImporte();
		}
		return total;
	}

}
```

```
package com.arquitecturajava.ejemplo2;

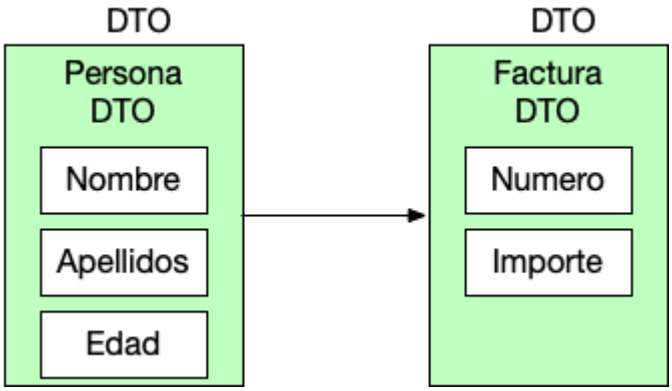
public class Linea {

	private int numero;
```

```
        private String concepto;
        private int importe;
        public int getNumero() {
                return numero;
        }
        public void setNumero(int numero) {
                this.numero = numero;
        }
        public String getConcepto() {
                return concepto;
        }
        public void setConcepto(String concepto) {
                this.concepto = concepto;
        }
        public int getImporte() {
                return importe;
        }
        public void setImporte(int importe) {
                this.importe = importe;
        }
        public Linea(int numero, String concepto, int importe) {
                super();
                this.numero = numero;
                this.concepto = concepto;
                this.importe = importe;
        }
}
```

Data Transfer Objects y Agregados

Nosotros lo que vamos a querer es construir un DTO que contenga una parte de la información de la Persona y otra parte de la información de las Facturas y sus Lineas.



Esta operación ya no es tan trivial y tendremos que construir un DTO bastante especifico con una estructura tipo agregado que contenga eso sí la información pura que nosotros necesitamos .

```
package com.arquitecturajava.ejemplo2;

import java.util.ArrayList;
import java.util.List;

public class PersonaFacturaDTO {

	private String nombre;
	private String apellidos;
	private int edad;
	private List<FacturaDTO> facturas= new
ArrayList<FacturaDTO>();
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public String getApellidos() {
```

```
                return apellidos;
        }
        public void setApellidos(String apellidos) {
                this.apellidos = apellidos;
        }
        public int getEdad() {
                return edad;
        }
        public void setEdad(int edad) {
                this.edad = edad;
        }
        public void addFactura(FacturaDTO dto) {
                facturas.add(dto);
        }
        public List<FacturaDTO> getFacturas() {
                return facturas;
        }
        public void setFacturas(List<FacturaDTO> facturas) {
                this.facturas = facturas;
        }
}
```

Como podemos ver se trata de una PersonaFacturaDTO pero esta Persona contiene solo una parte muy concreta de la información de las Facturas . El numero y el importeTotal , no contiene las lineas en sí .

```
package com.arquitecturajava.ejemplo2;

public class FacturaDTO {

        private int numero;
        private int importeTotal;
```

```
	public int getNumero() {
		return numero;
	}


	public void setNumero(int numero) {
		this.numero = numero;
	}


	public int getImporteTotal() {
		return importeTotal;
	}


	public void setImporteTotal(int importeTotal) {
		this.importeTotal = importeTotal;
	}


	public FacturaDTO(int numero, int importeTotal) {
		super();
		this.numero = numero;
		this.importeTotal = importeTotal;
	}
}
```

De esta manera cuando transformemos el grafo de Persona ->Factura->Linea en PersonaDTO->FacturaDTO. Tendremos que realizar una transformación bastante manual.

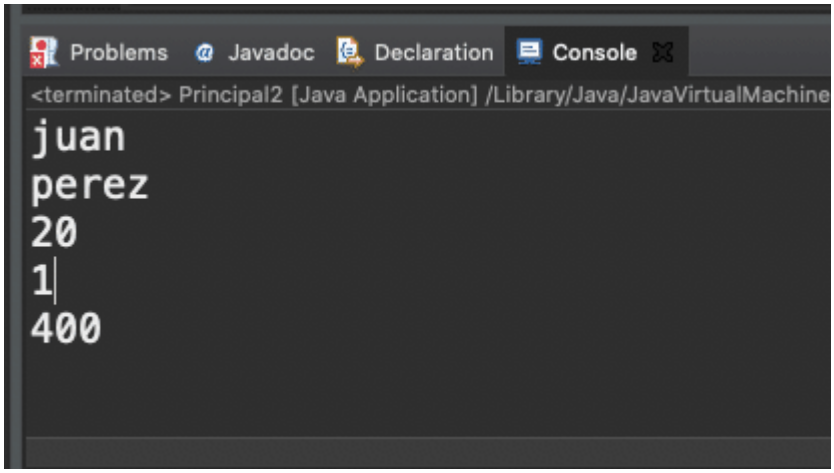
```
package com.arquitecturajava.ejemplo2;
```

```
public class Principal {

	public static void main(String[] args) {
		Persona p= new Persona("juan","perez",20);
		Factura f= new Factura(1,"ordenador");
		Linea linea1= new Linea(1,"cpu",200);
		Linea linea2= new Linea(2,"monitor",200);
		f.addLinea(linea1);
		f.addLinea(linea2);
		p.addFactura(f);
		PersonaFacturaDTO dto= new PersonaFacturaDTO();
		dto.setNombre(p.getNombre());
		dto.setApellidos(p.getApellidos());
		dto.setEdad(p.getEdad());
		for (Factura factura: p.getFacturas()) {
			dto.addFactura(new
FacturaDTO(factura.getNumero(),factura.getImporteTotal()));
		}
		System.out.println(dto.getNombre());
		System.out.println(dto.getApellidos());
		System.out.println(dto.getEdad());
		for (FacturaDTO fdto:dto.getFacturas()) {
			System.out.println(fdto.getNumero());
			System.out.println(fdto.getImporteTotal());
		}
	}

}
```

Este ejemplo si lo ejecutamos veremos como la información que necesitamos ha pasado a encontrarse dentro de un DTO.



DTO Assembler y el principio DRY

Para evitar la repetición de código , acabaremos construyendo una clase especifica que se encargue de este tipo de operaciones.

```
package com.arquitecturajava.ejemplo2;

public class PersonaFacturaAssembler {

	public static PersonaFacturaDTO personaToDTO(Persona persona)
{
		PersonaFacturaDTO dto= new PersonaFacturaDTO();
		dto.setNombre(persona.getNombre());
		dto.setApellidos(persona.getApellidos());
		dto.setEdad(persona.getEdad());
		for (Factura factura: persona.getFacturas()) {
			dto.addFactura(new
FacturaDTO(factura.getNumero(),factura.getImporteTotal()));
		}
		return dto;
	}
}
```

De esta manera podemos organizar el programa mejor y usar el DTO Assembler.

```
package com.arquitecturajava.ejemplo2;

public class Principal2 {

	public static void main(String[] args) {
		Persona p= new Persona("juan","perez",20);
		Factura f= new Factura(1,"ordenador");
		Linea linea1= new Linea(1,"cpu",200);
		Linea linea2= new Linea(2,"monitor",200);
		f.addLinea(linea1);
		f.addLinea(linea2);
		p.addFactura(f);
		PersonaFacturaDTO dto= new PersonaFacturaDTO();
		dto= PersonaFacturaAssembler.personaToDTO(p);
		imprimir(dto);
	}

	private static void imprimir(PersonaFacturaDTO dto) {
		System.out.println(dto.getNombre());
		System.out.println(dto.getApellidos());
		System.out.println(dto.getEdad());
		for (FacturaDTO fdto:dto.getFacturas()) {
			System.out.println(fdto.getNumero());
			System.out.println(fdto.getImporteTotal());
		}
	}

}
```

Un ejemplo de la automatización de la construcción de DTO Assemblers es MapStruts

Otros artículos relacionados

- Entity to DTO y Java 8
- REST DTO y JSON Arquitecturas Web y objetos
- JPA DTO (Data Transfer Object) y JPQL
- Curso Arquitectura Java