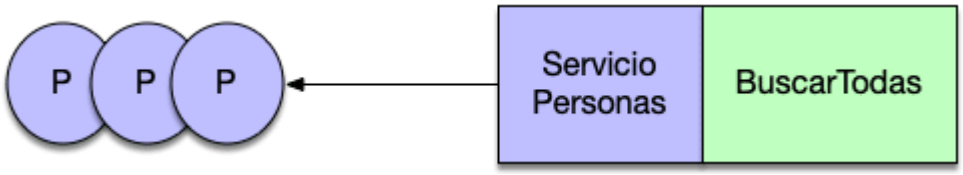
El uso de DTO o Data Transfer Object es uno de los conceptos más habituales a nivel de Arquitectura cuando devolvemos en nuestros servicios estructuras de datos . Muchos servicios devuelven objetos de negocio o gráfos con objetos de negocio relacionados . Es decir cuando nosotros tenemos un método que nos devuelve información sobre Personas lo más sencillo es devolver por ejemplo una lista de Personas desde un servicio.



Vamos a ver un ejemplo con la clase Persona:

CURSO SPRING REST

GRATIS

APUNTATE!!

```
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;
	}
}
```

Creamos una clase de servicio que devuelva una lista de Personas .

```
package com.arquitecturajava.ejemplo1;

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

public class ServicioPersonas {

	public List<Persona> buscarTodos() {
		Persona p1= new Persona("pedro","gomez",20);
		Persona p2= new Persona("angel","sanchez",20);
		Persona p3= new Persona("ana","perez",30);
		List<Persona> lista= new ArrayList<Persona>();
```

```
                lista.add(p1);
                lista.add(p2);
                lista.add(p3);
                return lista;
        }
}
```

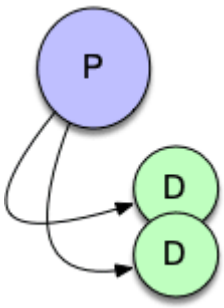
Usamos un programa main para imprimir la información . Este es el caso más sencillo a nivel de objetos de negocio.

```
package com.arquitecturajava.ejemplo1;

import java.util.List;

public class Principal {

        public static void main(String[] args) {
                ServicioPersonas servicio= new ServicioPersonas();
                List<Persona> personas= servicio.buscarTodos();
                for (Persona persona: personas) {
                        System.out.println(persona.getNombre());
                }

        }

}
```

Esta siempre debe ser nuestra primera opción , si es posible simplemente devolver los objetos de negocio . Lamentablemente , las cosas no son siempre tan sencillas .

Java Data Transfer Object y Grafos

La segunda opción posible es devolver un gráfo de objetos de tal manera que podamos devolver estructuras más complejas con información adicional. Por ejemplo una Persona puede realizar varios deportes :



Vamos a ver su código fuente :

TODOS LOS CURSOS
PROFESIONALES
25$/MES
APUNTATE!!

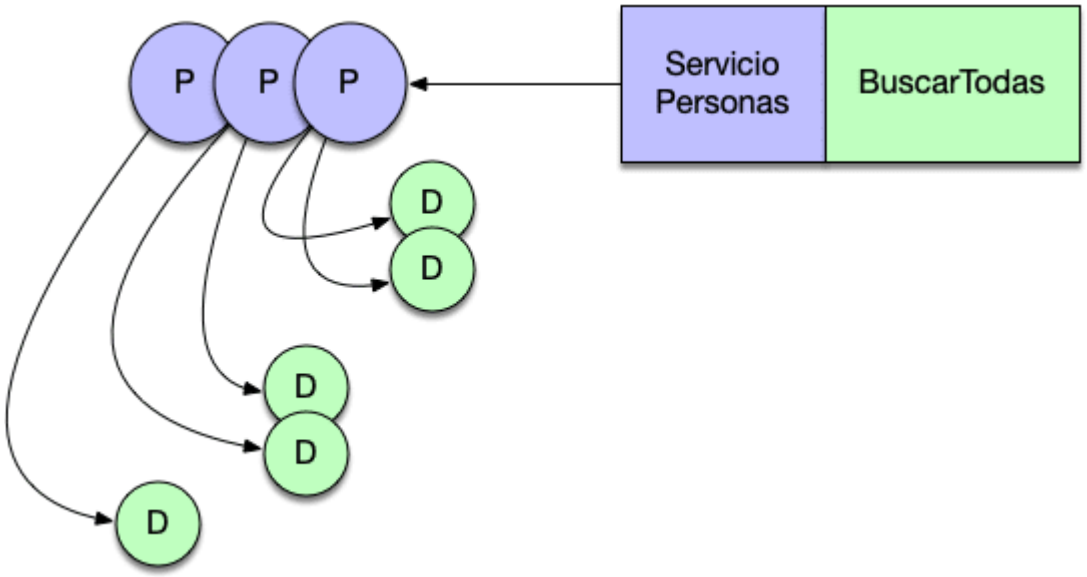
```
package com.arquitecturajava.ejemplo2;

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

public class Persona {

        private String nombre;
        private String apellidos;
```

```
private int edad;
public List<Deporte> getDeportes() {
        return deportes;
}
public void setDeportes(List<Deporte> deportes) {
        this.deportes = deportes;
}
private List<Deporte> deportes= new ArrayList<Deporte>();
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 void addDeporte(Deporte deporte) {
                this.deportes.add(deporte);
        }
}
```

```
package com.arquitecturajava.ejemplo2;

public class Deporte {

        private String nombre;
        private int nivel;
        public String getNombre() {
                return nombre;
        }
        public void setNombre(String nombre) {
                this.nombre = nombre;
        }
        public int getNivel() {
                return nivel;
        }
        public void setNivel(int nivel) {
                this.nivel = nivel;
        }
        public Deporte(String nombre, int nivel) {
                super();
                this.nombre = nombre;
                this.nivel = nivel;
        }

}
```

En este caso la clase Persona contiene un ArrayList de deportes . Es momento de modificar la clase de Servicio para que nos devuelve una estructura agregada.



Vamos a ver su código:

```
package com.arquitecturajava.ejemplo2;

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

public class ServicioPersonas {

        public List<Persona> buscarTodos() {
                Persona p1= new Persona("pedro","gomez",20);
                Persona p2= new Persona("angel","sanchez",20);
                Persona p3= new Persona("ana","perez",30);
                p1.addDeporte(new Deporte("futbol",1));
                p1.addDeporte(new Deporte("tenis",2));
                p2.addDeporte(new Deporte("basket",2));
```

```
                p2.addDeporte(new Deporte("padel",1));
                p3.addDeporte(new Deporte("basket",2));
                List<Persona> lista= new ArrayList<Persona>();
                lista.add(p1);
                lista.add(p2);
                lista.add(p3);
                return lista;
        }
}
```

Los Grafos nos ayudan a solventar muchos de los problemas que tenemos a la hora de relacionar información . Sin embargo existen situaciones en las que los grafos nos devuelven demasiada información y en una pantalla concreta necesitamos una información concreta . Es aquí donde un DTO (Data Transfer Object) nos ayudan sobre manera ya que nos permiten crear una clase que contenga únicamente los atributos necesarios.

```
package com.arquitecturajava.ejemplo3;

public class PersonaDeporteDTO {

        private String nombre;
        private String deporte;
        public String getNombre() {
                return nombre;
        }
        public void setNombre(String nombre) {
                this.nombre = nombre;
        }
        public String getDeporte() {
                return deporte;
        }
        public void setDeporte(String deporte) {
```

```
                this.deporte = deporte;
        }
        public PersonaDeporteDTO(String nombre, String deporte) {
                super();
                this.nombre = nombre;
                this.deporte = deporte;
        }
}
```

En este caso hemos construido una clase que únicamente devuelve dos propiedades que son las que nosotros necesitaremos en la pantalla.


```
package com.arquitecturajava.ejemplo3;

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

public class ServicioPersonas {

        public List<PersonaDeporteDTO> buscarTodos() {

                List<PersonaDeporteDTO> lista = new
ArrayList<PersonaDeporteDTO>();

                lista.add(new PersonaDeporteDTO("pedro", "futbol"));

                lista.add(new PersonaDeporteDTO("pedro", "tenis"));
```

```
                lista.add(new PersonaDeporteDTO("angel", "basket"));

                lista.add(new PersonaDeporteDTO("angel", "padel"));
                lista.add(new PersonaDeporteDTO("ana", "basket"));

                return lista;

        }
}
```

El uso de Data Transfer Objects nos simplifica el trabajo y muchas veces también mejora en rendimiento a traves de http ya que enviamos menos enformación. Cada una de las opciones aporta sus cosas . El devolver objetos de negocio puros , es sencillo el devolver gráfos evita que tengamos que construir DTOS , pero a cambio nos devuelve toda la información . Finalmente el uso de DTOS genera nuevas clases pero nos permite ajustarnos de una manera muy directa a nuestras necesidades . No hay una solución mejor que otra cada casuística se ajusta mejor a una de las tres soluciones.

Otros artículos relacionados

CURSO Introducción Spring Data

GRATIS

APUNTATE!!

- JPA DTO
- Entity to DTO y Java 8
- El concepto de API GateWay Pattern
- DTO