La anotación @Basic de JPA es una de esas anotaciones que es importante pero que muchas veces tenemos olvidadas. Vamos a echarla un vistazo. Supongamos que tenemos una clase Cliente que persistimos en una base de datos. La clase tiene la siguiente estructura en Java.

```
package com.arquitecturajava.jpa.bo;

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

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;

@Entity
@Table(name="Clientes")
public class Cliente {
	@Id
	private String dni;
	private String nombre;
	private String apellidos;
	private int edad;
	private int telefono;
	@OneToMany(mappedBy="cliente")
	private List<Factura> listaFacturas= new ArrayList<Factura>();
	public List&lt;Factura&gt; getListaFacturas() {
		return listaFacturas;
	}
```

```
	public void setListaFacturas(List&lt;Factura&gt;
listaFacturas) {
		this.listaFacturas = listaFacturas;
	}


	public Cliente(String dni, String nombre, String apellidos,
int edad, int telefono) {
		super();
		this.dni = dni;
		this.nombre = nombre;
		this.apellidos = apellidos;
		this.edad = edad;
		this.telefono = telefono;
	}
	public Cliente(String dni) {
		super();
		this.dni = dni;
	}


	public Cliente() {
		super();
	}


	public String getDni() {
		return dni;
	}
	public void setDni(String dni) {
		this.dni = dni;
```

```
	}
	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 int getTelefono() {
		return telefono;
	}
	public void setTelefono(int telefono) {
		this.telefono = telefono;
	}
}
```

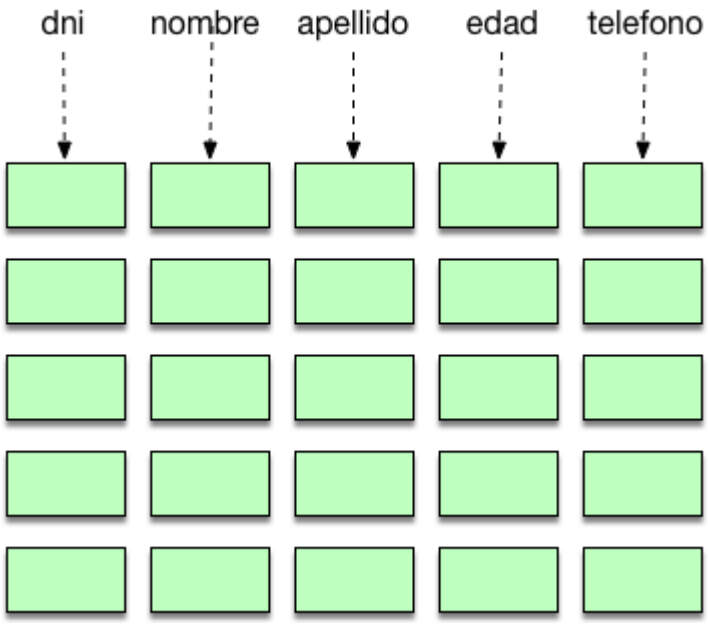
Hasta aquí todo es muy clásico , si realizamos una consulta de JPA QL y seleccionamos los clientes de la base de datos el código será el siguiente:

```
package com.arquitecturajava.jpa;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.TypedQuery;

import com.arquitecturajava.jpa.bo.Cliente;

public class Principal002JPAQL {

	public static void main(String[] args) {

		EntityManagerFactory emf;
		emf=Persistence.createEntityManagerFactory("jpa");
		EntityManager em= emf.createEntityManager();
		TypedQuery<Cliente> consulta=em.createQuery("select  c
from Cliente c",Cliente.class);
		List<Cliente> lista= consulta.getResultList();
		for(Cliente c:lista) {
			System.out.printf(" %s %s %s %s
%n",c.getDni(),c.getNombre(),c.getApellidos(),c.getEdad());

		}
		em.close();

	}

}
```

El resultado nos lo imprimirá en la consola:

```
124 manuel alvarez 50
234 pedro perez 20
456 gema blanco 20
567 vanesa perez 20
667 mar lopez 50
789 alfonso diaz 40
```

¿Es correcta la consulta JPA que hemos construido ? la realidad es que sí. Acabamos de seleccionar todos los Clientes de la base de datos.

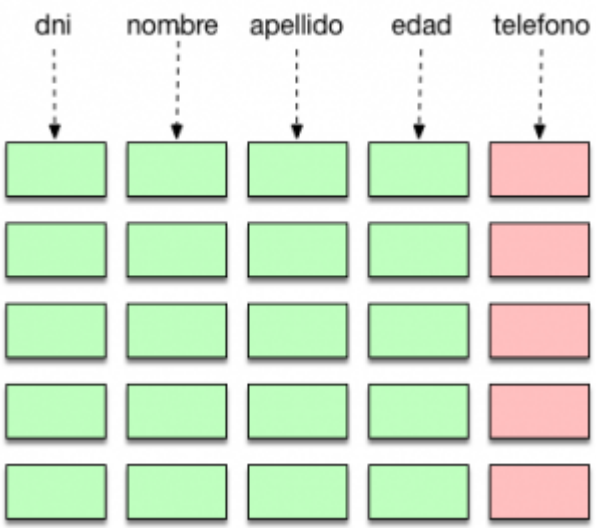


La consulta en la consola es:

```
-SELECT DNI, APELLIDOS, EDAD, NOMBRE, TELEFONO FROM Clientes
```

Usando JPA @Basic

Sin embargo hay algunas situaciones en las que deseamos afinar más la consulta y no seleccionar una columna porque por ejemplo sea un campo block que guarda un PDF .



¿Cómo se puede hacer esto? . En nuestro caso vamos a suponer que ese campo es el teléfono . Queremos indicarle a JPA que cuando realizemos una consulta a la tabla este campo no se seleccione por defecto. Para ello usaremos la anotación @Basic . El primer paso es modificar nuestra Entity y añadir la anotación @Basic.

```
 @Basic(fetch=FetchType.LAZY)
private int telefono;
```

Es momento de volver a ejecutar nuestra consulta de JPA y ver los cambios

```
124 manuel alvarez 50
234 pedro perez 20
456 gema blanco 20
567 vanesa perez 20
667 mar lopez 50
789 alfonso diaz 40
```

Curiosamente no existe ningún cambio los datos vuelven a ser impresos y punto. Ahora bien recordemos que los datos que se imprimen no incluyen el telefono. Sin embargo la consulta anterior si incluía ese campo aunque no lo presentara en pantalla. La nueva consulta que acabamos de ejecutar no incluye ese campo

```
-SELECT DNI, APELLIDOS, EDAD, NOMBRE FROM Clientes
```

Acabamos de usar la anotación @Basic para indicarle a JPA que no debe seleccionar nunca por defecto esa columna . Solamente obtendremos información sobre ella cuando accedamos al campo de forma directa.

Otros artículos relacionados:

1. Java Generic Repository y JPA
2. JPA Criteria API , un enfoque diferente
3. JPA @GeneratedValue y Primary keys
4. Un ejemplo de JPA Entity Graph

Recursos Externos:

1. JPA Wikipedia