El concepto de Java Tuple , es un concepto un poco peculiar . Otros lenguajes como C# o TypeScript lo incorporan de forma natural pero en Java es necesario usar una librería adicional Java Tuples . Esta librería nos la podemos importar con maven ya que se trata de un añadido muy sencillo al lenguaje Java.

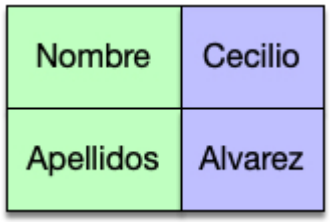
```
<!-- https://mvnrepository.com/artifact/org.javatuples/javatuples -->
<dependency>
    <groupId>org.javatuples</groupId>
    <artifactId>javatuples</artifactId>
    <version>1.2</version>
</dependency>
```

¿Para que sirve un Java Tuple comparado con un concepto más clásico de diccionario o HashMap. Vamos a ver un ejemplo usando un diccionario :

```
package com.arquitecturajava;

import java.util.HashMap;
import java.util.Map;

public class Principal {

  public static void main(String[] args) {

    Map<String,String> mapa= new HashMap<>();
    mapa.put("nombre","cecilio");
    mapa.put("apellidos","alvarez");
    System.out.println(mapa.get("nombre"));
    System.out.println(mapa.get("apellidos"));

  }
```

```
}
```

En este caso hemos construido un diccionario de cadenas . El diccionario tiene dos items uno que define el nombre y su valor , en este caso nombre=”cecilio” y el otro el que define el apellido apellidos=”alvarez” . Por lo tanto estamos ante una estructura de clave/valor. Estas estructuras son muy útiles y en el mundo Java las necesitamos siempre.



Sin embargo hay situaciones que necesitamos otro tipo de estructura. Cuando uno maneja un diccionario siempre esta obligado a saber las claves para poder obtener los valores. Existen situaciones en las que no necesitamos las capacidades de un diccionario ya que por ejemplo sabemos el orden de los valores que vamos a recibir. Esto puede sonar un poco extraño pero vamos a ver un ejemplo que ayude a clarificar. Imaginemonos que necesitamos realizar una consulta contra la base de datos en esa consulta podemos elegir la columna que queremos y pasarle un valor . La base de datos nos devolverá el resultado cuando generemos correctamente la consulta. Una opción parar realizar esto es usar un diccionario.

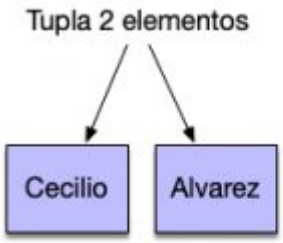
```
package com.arquitecturajava;

import java.util.HashMap;
import java.util.Map;

public class Principal3 {

  public static void main(String[] args) {

    Map<String,String> mapa= new HashMap<>();
```

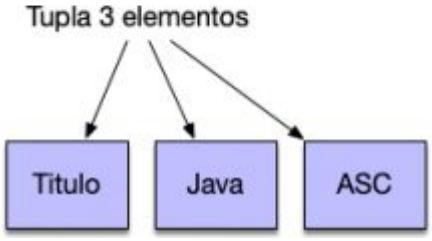
```
    mapa.put("columna","titulo");
    mapa.put("valor","java");
    String consulta= "select "+mapa.get("columna")+  " from libros
where " + mapa.get("columna") +"='"+ mapa.get("valor")+"'";
    System.out.println(consulta);
  }

}
```

Si ejecutamos este código nos podremos dar cuenta que se imprime la consulta por la consola:

```
select titulo from libros where titulo='java'
```

¿Es esto totalmente correcto? En principio lo parece , pero claro mucha gente nos dirá . Uff usar un diccionario para simplemente acceder a un conjunto de valores … que no necesitan de ningún tipo de filtrado , quizás podamos hacer algo más directo. Para esto existe el concepto de tupla .





Una tupla no almacena clave y valores almacena solo valores.

```
package com.arquitecturajava;

import org.javatuples.Pair;

public class Principal2 {

  public static void main(String[] args) {

    Pair<String, String> tupla = new Pair<String, String>("titulo",
"java");

    String consulta = "select " + tupla.getValue0() + " from libros
where " + tupla.getValue1() + "='"+ tupla.getValue1() + "'";
    System.out.println(consulta);

  }

}
```

En este caso hemos usado una java tuple en formato de pareja ya que es la más habitual pero se puede ver cómo el código se simplifica claramente . Las tuplas pueden tener hasta 10 elementos como valores por lo tanto aportan bastante flexibilidad al código.

Otros artículos relacionados

1. Java List to Map y el uso de Collectors
2. Java Collector Join Streams y Strings
3. Java Collections List vs Set (I)