Tabla de Contenidos

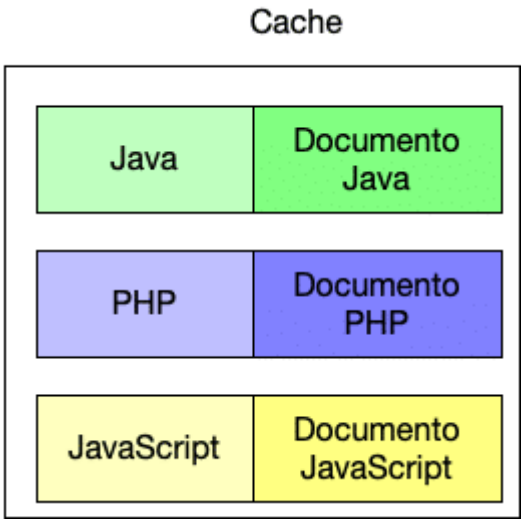
El uso de Java Map computeIfAbsent es algo de lo más común en Java ya que todos usamos diccionarios cuando programamos pero es algo poco conocido . Los Mapas se usan habitualmente para almacenar parejas de claves y valores y en muchos de estos casos los mapas valen para almacenar valores como si se tratara de una cache veamos un ejemplo sencillo a partir de la clase Documento.

```
package com.arquitecturajava;

public class Documento {

	private String nombre;
	private String contenido;
	public Documento(String nombre, String contenido) {
		super();
		this.nombre = nombre;
		this.contenido = contenido;
	}
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public String getContenido() {
		return contenido;
```

```
        }
        public void setContenido(String contenido) {
                this.contenido = contenido;
        }
}
```

Caches y Documentos

Una vez construida la clase documento es momento de usar esta clase dentro de una pequeña cache de tal manera que si no tenemos en cache el documento nos lo añada a la cache que es un Java Map.



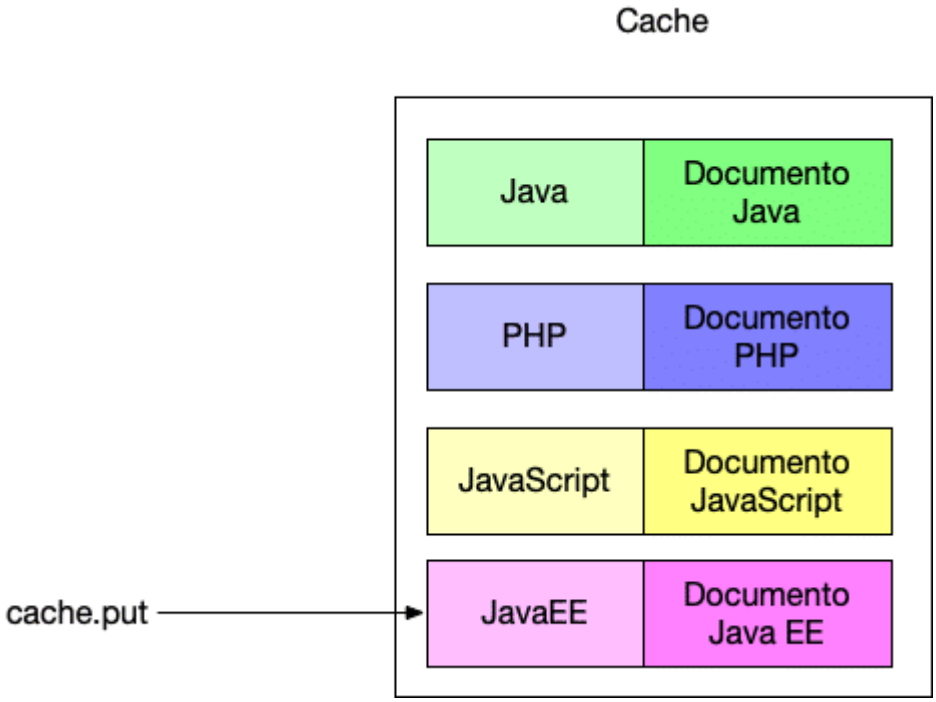
Vamos a verlo en código:

```
package com.arquitecturajava;

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

public class Principal {
```

```
        static Map<String, Documento> cache = new HashMap<>();

        public static void main(String[] args) {

                cargarCache();
                String nombre = "javaee";
                String contenido = "java ee es una plataforma";

                if (cache.get(nombre) == null) {

                        Documento d = new Documento(nombre,
contenido);
                        cache.put(nombre, d);
                }

                Documento otro = cache.get("javaee");
                System.out.println(otro.getNombre());
                System.out.println(otro.getContenido());
                System.out.println(cache.size());

        }

        public static Map<String, Documento> cargarCache() {

                cache.put("java", new Documento("java", "java es un
lenguaje de programacion"));
                cache.put("php", new Documento("php", "php nos
programa la web"));
                cache.put("javascript", new Documento("javascript",
"es un lenguaje sencillo"));
                return cache;
```

```
        }

}
```

En este caso hemos usado el Java Map como cache y hemos comprobado que no existía un documento antes de añadirlo a la cache.



Ejecutamos el programa y el resultado le podemos ver por la consola.

Vemos que el mapa que hace las funciones de cache y dispone ya dentro de él de 4 elementos los 3 anteriores y el nuevo. Todo muy sencillo de construir. Lamentablemente esta es una operativa que es muy común y continuamente nos estaremos encontrando con el siguiente bloque de código para comprobar si un elemento esta o no en la cache antes de

añadirle.

```
if (cache.get(nombre) == null) {

                        Documento d = new Documento(nombre,
contenido);
                        cache.put(nombre, d);
                }
```

Java Map ComputeIfAbsent

No parece muy complejo pero siempre es un poco perdida de tiempo . A partir de Java 8 estas 3 lineas de código se pueden substituir por el uso de Java Map ComputeIfAbsent

```
        cache.computeIfAbsent(nombre, k-> new Documento(k,contenido));
```

Con lo cual el código se quedaría en lo siguiente:

```
package com.arquitecturajava;

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

public class PrincipalCompute {

        static Map<String, Documento> cache = new HashMap<>();

        public static void main(String[] args) {

                cargarCache();
                String nombre = "javaee";
```

```
                String contenido = "java ee es una plataforma";
                //simplificado
                cache.computeIfAbsent(nombre, k-> new
Documento(k,contenido));

                Documento otro = cache.get("javaee");
                System.out.println(otro.getNombre());
                System.out.println(otro.getContenido());
                System.out.println(cache.size());

        }

        public static Map<String, Documento> cargarCache() {

                cache.put("java", new Documento("java", "java es un
lenguaje de programacion"));
                cache.put("php", new Documento("php", "php nos
programa la web"));
                cache.put("javascript", new Documento("javascript",
"es un lenguaje sencillo"));
                return cache;

        }

}
```

Conclusión

Este es algo mucho mas compacto y rápido de implementar.

Otros artículos relacionados

- Java List to Map y el uso de Collectors

- Java ArrayList for y sus opciones
- Java 9 Factory Methods (List,Set,Map)
- Java Generic Repository y JPA
- Java Generic Methods
- Curso Java APIS