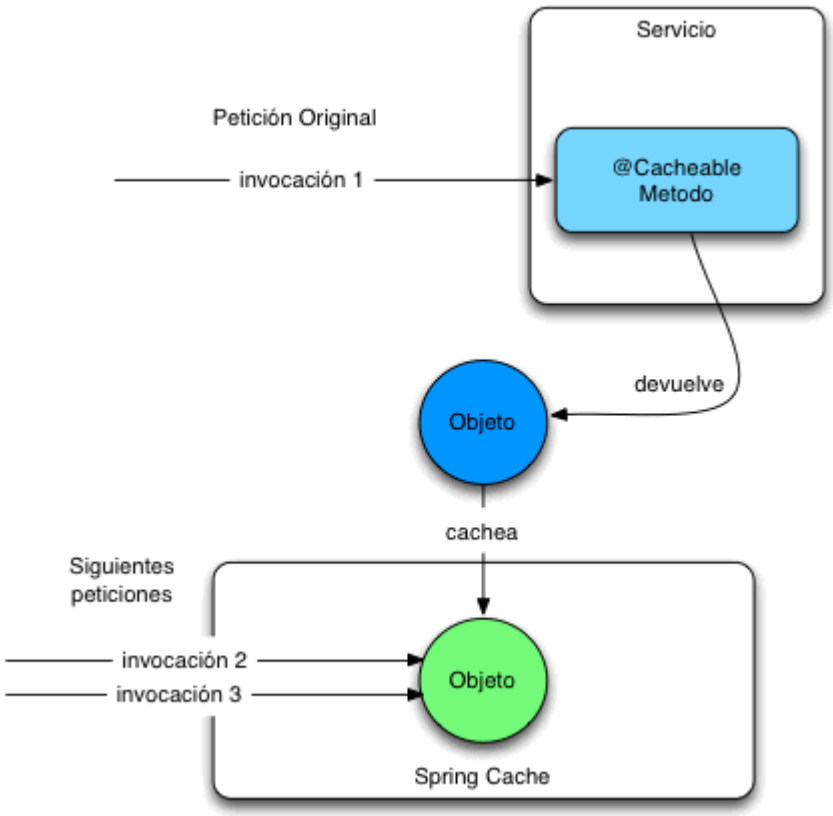
Acabamos de ver como usar Java equals y hashcode

Spring Cache es una de las características de Spring Framework que nos puede sacar de más de un apuro. Habitualmente usamos Spring para crear Servicios y Repositorios que definen la parte del Modelo de nuestra aplicación. En bastantes casos nos encontramos con situaciones en las que un Servicio siempre devuelve la misma información, por ejemplo tablas paramétricas. Es en este tipo de situaciones en las que no tiene sentido estar continuamente realizando una consulta a la base de datos ,la información no va a modificarse. Para solventar este tipo de problemas Spring aporta soluciones de Cache que permiten almacenar en memoria datos devueltos por un método concreto.



Vamos a ver el ejemplo de hola mundo para hacernos una idea. Para ello añadiremos los siguientes artefactos a nuestro proyecto Maven.

```
&amp;lt;project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd"&amp;gt;
 &amp;lt;modelVersion&amp;gt;4.0.0&amp;lt;/modelVersion&amp;gt;
 &amp;lt;groupId&amp;gt;com.arquitecturava&amp;lt;/groupId&amp;gt;
 &amp;lt;artifactId&amp;gt;SpringCache&amp;lt;/artifactId&amp;gt;
 &amp;lt;version&amp;gt;0.0.1-SNAPSHOT&amp;lt;/version&amp;gt;
 &amp;lt;name&amp;gt;SpringCache&amp;lt;/name&amp;gt;
 &amp;lt;dependencies&amp;gt;
 &amp;lt;dependency&amp;gt;
 &amp;lt;groupId&amp;gt;org.springframework&amp;lt;/groupId&amp;gt;
 &amp;lt;artifactId&amp;gt;spring-core&amp;lt;/artifactId&amp;gt;
 &amp;lt;version&amp;gt;4.2.1.RELEASE&amp;lt;/version&amp;gt;
 &amp;lt;/dependency&amp;gt;
 &amp;lt;dependency&amp;gt;
 &amp;lt;groupId&amp;gt;org.springframework&amp;lt;/groupId&amp;gt;
 &amp;lt;artifactId&amp;gt;spring-beans&amp;lt;/artifactId&amp;gt;
 &amp;lt;version&amp;gt;4.2.1.RELEASE&amp;lt;/version&amp;gt;
 &amp;lt;/dependency&amp;gt;

 &amp;lt;dependency&amp;gt;
 &amp;lt;groupId&amp;gt;org.springframework&amp;lt;/groupId&amp;gt;
 &amp;lt;artifactId&amp;gt;spring-context&amp;lt;/artifactId&amp;gt;
 &amp;lt;version&amp;gt;4.2.1.RELEASE&amp;lt;/version&amp;gt;
 &amp;lt;/dependency&amp;gt;

 &amp;lt;/dependencies&amp;gt;
&amp;lt;/project&amp;gt;
```

El siguiente paso es definir el contenido del applicationContext.xml:

```
&amp;lt;beans xmlns="http://www.springframework.org/schema/beans"

 xmlns:context="http://www.springframework.org/schema/context"
 xmlns:mvc="http://www.springframework.org/schema/mvc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:p="http://www.springframework.org/schema/p"
 xmlns:cache="http://www.springframework.org/schema/cache"

 xsi:schemaLocation="
 http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
 http://www.springframework.org/schema/context
 http://www.springframework.org/schema/context/spring-context-4.0.xsd
 http://www.springframework.org/schema/cache
 http://www.springframework.org/schema/cache/spring-cache.xsd"&gt;

 &amp;lt;cache:annotation-driven /&amp;gt;

 &amp;lt;context:component-scan base-package="com.arquitecturajava"
/&amp;gt;
 &amp;lt;bean id="cacheManager"
class="org.springframework.cache.concurrent.ConcurrentMapCacheManager"
&amp;gt;&amp;lt;/bean&amp;gt;

&amp;lt;/beans&amp;gt;
```

En este fichero se ha dado de alta uno de los tipos de Cache de SpringFramework , que pasaremos a utilizar en un Servicio.

```
package com.arquitecturajava;

import java.time.LocalDateTime;

import org.springframework.cache.annotation.Cacheable;
import org.springframework.stereotype.Service;

@Service
public class Servicio {
 public LocalDateTime fechaActual() {

 return LocalDateTime.now();

 }
}
```

El servicio nos devuelve la fecha y hora del sistema actual , vamos a invocarlo desde una aplicación de consola.

```
package com.arquitecturajava;

import
org.springframework.context.support.ClassPathXmlApplicationContext;

public class Principal {
```

```
 public static void main(String[] args) {

 ClassPathXmlApplicationContext contexto= new
ClassPathXmlApplicationContext("applicationContext.xml");
 Servicio s=contexto.getBean(Servicio.class);

 Thread t = new Thread (() -&amp;gt; {
 try {

 for (int i = 0; i &amp;lt; 10; i++) {
 Thread.sleep(1000);
 System.out.println(s.fechaActual());
 }

 } catch (Exception e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
 }

 });
 t.start();
 }

}
```

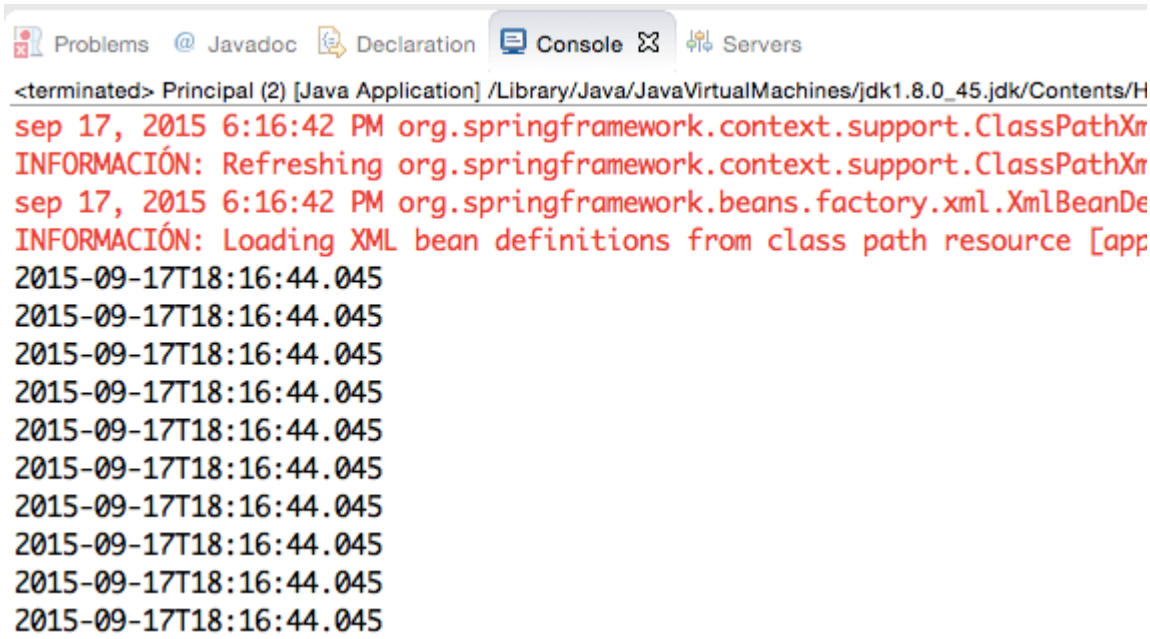
El código del cliente invoca 10 veces al servicio y descansa 1 segundo en cada invocación. La consola muestra lo siguiente:

```
sep 17, 2015 6:12:03 PM org.springfr
INFORMACIÓN: Refreshing org.springfr
sep 17, 2015 6:12:03 PM org.springfr
INFORMACIÓN: Loading XML bean defini
2015-09-17T18:12:05.767
2015-09-17T18:12:06.776
2015-09-17T18:12:07.781
2015-09-17T18:12:08.781
2015-09-17T18:12:09.783
2015-09-17T18:12:10.788
2015-09-17T18:12:11.793
2015-09-17T18:12:12.798
2015-09-17T18:12:13.802
2015-09-17T18:12:14.803
```

La información devuelta cambia continuamente, es momento de activar la cache para el método utilizando la anotación de Spring Cache (@Cacheable) a nivel de método.

```
package com.arquitecturajava;

import java.time.LocalDateTime;

import org.springframework.cache.annotation.Cacheable;
import org.springframework.stereotype.Service;

@Service
public class Servicio {

@Cacheable("fecha")
public LocalDateTime fechaActual() {

return LocalDateTime.now();
```

```
}
}
```



Los datos han quedado cacheados de forma permanente. Spring Cache soporta muchas configuraciones y sistemas de cache diferentes que nos pueden ayudar aumentar el rendimiento de la aplicación.

Otros artículos relacionados:

Spring vs EJB

Spring Anotaciones vs XML

Spring Framework y Aislamiento