**UsingTriggersRemove**

*How to use the @TriggersRemove annotation.*   
[Featured](https://code.google.com/p/ehcache-spring-annotations/w/list?q=label:Featured)

Updated Sep 30, 2010 by [eric.dalquist](https://code.google.com/u/eric.dalquist/)

**Using @TriggersRemove**

ehcache-spring-annotations includes another annotation - @TriggersRemove - which can be applied to methods in order to trigger removal of a single element or optionally of all elements from one or more caches. Please read the documentation for [@Cacheable](https://code.google.com/p/ehcache-spring-annotations/wiki/UsingCacheable) before reading this page.

**Default Example**

In the preceding example, the removeAll option is enabled and all Elements from the cache are removed.

The default behavior of the @TriggersRemove will only remove one Element upon invocation.

Consider our example again:

public class DefaultWeatherDao implements WeatherDao {  
      
    @Cacheable(cacheName="weatherCache")  
    public Weather getWeather(String zipCode);  
      
    @Cacheable(cacheName="locationSearchCache")  
    public List<Location> findLocations(String locationSearch);  
}

Say we want to add an delete method to this class. It will contain some logic to remove the Weather object from persistence. Ideally, we'd like to use @TriggersRemove on this new delete method to remove the corresponding element from the cache automatically upon invocation:

public class DefaultWeatherDao implements WeatherDao {  
      
    @Cacheable(cacheName="weatherCache")  
    public Weather getWeather(String zipCode);  
      
    @Cacheable(cacheName="locationSearchCache")  
    public List<Location> findLocations(String locationSearch);  
  
    public Weather deleteWeather(String zipCode);  
}

Simply adding @TriggersRemove(cacheName="weatherCache") to the deleteWeather function as defined above won't work. The[CacheKeyGenerator](https://code.google.com/p/ehcache-spring-annotations/wiki/CacheKeyGenerators) interface accepts the [MethodInvocation](http://aopalliance.sourceforge.net/doc/org/aopalliance/intercept/MethodInvocation.html) as it's sole argument - which by default includes the method arguments **and** method name. If we want invocation of the deleteWeather method to remove the same element out of the cache that was stored by getWeather, we need to make sure that the invocation of the deleteWeather and getWeather method results in the same calculated cache key.

In general terms, you have to configure an appropriate [CacheKeyGenerator](https://code.google.com/p/ehcache-spring-annotations/wiki/CacheKeyGenerators) such that the cache keys generated by your @Cacheable method will match to the cache keys generated by your @TriggersRemove method.

Example:

public class DefaultWeatherDao implements WeatherDao {  
      
    @Cacheable(cacheName="weatherCache",   
        keyGenerator = @KeyGenerator (  
                name = "HashCodeCacheKeyGenerator",  
                properties = @Property( name="includeMethod", value="false" )  
            )  
        )  
    )  
    public Weather getWeather(String zipCode);  
      
    @Cacheable(cacheName="locationSearchCache")  
    public List<Location> findLocations(String locationSearch);  
  
    @TriggersRemove(cacheName="weatherCache",   
        keyGenerator = @KeyGenerator (  
                name = "HashCodeCacheKeyGenerator",  
                properties = @Property( name="includeMethod", value="false" )  
            )  
        )  
    )  
    public Weather deleteWeather(String zipCode);  
}

With the includeMethod property set to false on a HashCodeCacheKeyGenerator instance, the method name will be ignored when generating cache Element keys. This instance will only use the method arguments. In this case, the getWeather and deleteWeather methods have the same type and number of arguments (a single String).

**removeAll Example**

The first example will demonstrate the removeAll option, which is **not** enabled by default.

Consider the original example from [UsingCacheAnnotations](https://code.google.com/p/ehcache-spring-annotations/wiki/UsingCacheAnnotations):

public class DefaultWeatherDao implements WeatherDao {  
      
    @Cacheable(cacheName="weatherCache")  
    public Weather getWeather(String zipCode);  
      
    @Cacheable(cacheName="locationSearchCache")  
    public List<Location> findLocations(String locationSearch);  
}

We can add a method to this class that triggers invocation of [Ehcache#removeAll()](http://ehcache.org/apidocs/net/sf/ehcache/Ehcache.html#removeAll()) each time it is executed:

public class DefaultWeatherDao implements WeatherDao {  
      
    @Cacheable(cacheName="weatherCache")  
    public Weather getWeather(String zipCode);  
      
    @Cacheable(cacheName="locationSearchCache")  
    public List<Location> findLocations(String locationSearch);  
  
    @TriggersRemove(cacheName="weatherCache", removeAll=true)  
    public void clearWeatherCache();  
}

The clearWeatherCache method doesn't have to have a single line of code. Upon invocation, all Elements inside the Ehcache named "weatherCache" will be removed. This example is sometimes useful in conjunction with a trigger for periodic execution.

**Multiple Caches Example**

The @TriggersRemove annotation can target multiple caches. Simply provide an array of cache names for the cacheName parameter and each will be affected. For **removeAll** each named cache will have all entries removed in order. If **removeAll** is not specified it is important to note that only a single [CacheKeyGenerator](https://code.google.com/p/ehcache-spring-annotations/wiki/CacheKeyGenerators) is used for all caches so they must share a common key generation strategy and key components for targeting multiple caches to be useful.

public class DefaultWeatherDao implements WeatherDao {  
      
    @Cacheable(cacheName="weatherCache")  
    public Weather getWeather(String zipCode);  
      
    @Cacheable(cacheName="locationSearchCache")  
    public List<Location> findLocations(String locationSearch);  
  
    @TriggersRemove(cacheName={"weatherCache", "locationSearchCache"}, removeAll=true)  
    public void clearCaches();  
}