EhCache

> EhCache 설정 및 사용 방법

Configurations

의존성 설정 (gradle)

- ~/build.gradle 파일에 아래 내용 추가

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-cache'
    // https://mvnrepository.com/artifact/org.ehcache/ehcache
    implementation group: 'org.ehcache', name: 'ehcache', version:
'3.10.0'
}
```

어플리케이션 설정

- src/main/resources/application.yml 파일에 설정 추가

```


spring:
  cache:
    jcache:
      config: classpath:ehcache.xml
```

- EhCache 사용 설정을 추가
- EhCache를 설정하는 파일 명을 ehcache.xml 로 지정

EHCACHE 설정

- src/main/resources/ehcache.xml 파일에 ehcache 설정 내용을 작성

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns='http://www.ehcache.org/v3'>
  <!-- 캐시를 저장할 디렉토리 위치 -->
    <persistence directory="${java.io.tmpdir}/ehcache-data/" />

  <!-- 캐시의 템플릿, 이 템플릿을 각각의 캐시에 적용 -->
    <cache-template name="default">
   <!-- 캐시 유지 시간 관리 -->
        <expiry>
            <tti unit="seconds">30</tti>
```

```
        </expiry>
    <!-- 캐시에 사용할 자원 정의 -->
        <resources>
            <!-- JVM heap 메모리, LRU -->
            <heap unit="MB">10</heap>
            <!-- JVM heap 메모리 외부의 메모리 -->
            <offheap unit="MB">50</offheap>
            <!-- disk 메모리, LFU -->
            <disk persistent="true" unit="GB">1</disk>
        </resources>
     </cache-template>

  <!-- 실제 적용할 캐시 정의 : exampleCache -->
     <cache alias="exampleCache" uses-template="default" >
    <!-- 사용할 키의 타입 정의 -->
        <key-type>java.lang.String</key-type>
    <!-- 캐시할 값의 타입 정의 -->
        <value-type>java.util.ArrayList</value-type>
     </cache>

  <!-- 실제 적용할 캐시 정의 : exampleCache2-->
     <cache alias="exampleCache2" uses-template="default" >
    <!-- 사용할 키의 타입 정의 -->
        <key-type>java.lang.String</key-type>
    <!-- 캐시할 값의 타입 정의 -->
        <value-type>java.lang.String</value-type>
     </cache></config>
```

- EhCache3 에는 defaultCache가 없고 cache-template을 적용하여 사용
- cache-template에 정의된 내용은 각 캐시에 적용되고, 각 캐시에서 재설정된 값이 있으면 개별 캐시에서 정의한 설정을 적용.
- 참고: 캐시 정책
  - LRU : 가장 오랫동안 호출 되지 않은 캐시를 삭제
  - LFU : 호출 빈도가 가장 적은 캐시를 삭제
  - FIFO : First In First Out, 캐시가 생성된 순서대로 가장 오래된 캐시를 삭제

EHCACHE 사용

Application에 캐시 사용 적용

- Application Main 클래스
  src/main/java/com/gsc/process/integration/GsxProcessIntegrationRunner.java

```
package com.gsc.process.integration;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
import org.springframework.cache.annotation.EnableCaching;

@SpringBootApplication
@EnableCaching
public class GscProcessIntegrationRunner {
  public static void main(String[] args) {
    System.out.println("Application for GSC Process Integration:
Started.");
    SpringApplication.run(GscProcessIntegrationRunner.class, args);
  }
}
```

- @EnableCaching: Cache 사용 어노테이션

Model 클래스 작성

```
@SuperBuilder
@Data
@EqualsAndHashCode(callSuper = false)
@NoArgsConstructor
@AllArgsConstructor
@ApiModel(description = "공통 코드 모델")
public class SampleApiModel implements Serializable {
    @ApiModelProperty("그룹 코드")
    private String grpCd;

    @ApiModelProperty("상세 코드")
    private String dtlCd;

    @ApiModelProperty("코드명")
    private String cdNm;

    @Data
    public static class Criteria {
        @ApiModelProperty("사용여부")
        private String useYn;
    }
}
```

- 캐시에 저장되는 모델 클래스는 반드시 Serializable 인터페이스를 구현해야 한다.

Service 클래스 작성

```
@Slf4j
@Service
public class SampleApiService {
   public List<SampleApiModel> getAll(SampleApiModel.Criteria criteria)
   {
       SampleApiModel model1 = new SampleApiModel("GRP_CD_000",
"DTL_CD_000", "CD_NM_000");
       SampleApiModel model2 = new SampleApiModel("GRP_CD_001",
"DTL_CD_001", "CD_NM_001");
       SampleApiModel model3 = new SampleApiModel("GRP_CD_002",
"DTL_CD_002", "CD_NM_002");
       SampleApiModel model4 = new SampleApiModel("GRP_CD_003",
"DTL_CD_003", "CD_NM_003");
       SampleApiModel model5 = new SampleApiModel("GRP_CD_004",
"DTL_CD_004", "CD_NM_004");

       List<SampleApiModel> list = new ArrayList<SampleApiModel>();

       list.add(model1);
       list.add(model2);
       list.add(model3);
       list.add(model4);
       list.add(model5);

       return list;
   }

   @Cacheable(cacheNames = "exampleCache", key="#useYn")
   public List<SampleApiModel> getCache(String useYn)  throws
InterruptedException
   {
       Thread.sleep(3000);    //    Code for TEST!

       SampleApiModel model1 = new SampleApiModel("GRP_CD_000",
"DTL_CD_000", "CD_NM_000");
       SampleApiModel model2 = new SampleApiModel("GRP_CD_001",
"DTL_CD_001", "CD_NM_001");
       SampleApiModel model3 = new SampleApiModel("GRP_CD_002",
"DTL_CD_002", "CD_NM_002");
       SampleApiModel model4 = new SampleApiModel("GRP_CD_003",
"DTL_CD_003", "CD_NM_003");
       SampleApiModel model5 = new SampleApiModel("GRP_CD_004",
"DTL_CD_004", "CD_NM_004");

       List<SampleApiModel> list = new ArrayList<SampleApiModel>();

       list.add(model1);
       list.add(model2);
       list.add(model3);
       list.add(model4);
       list.add(model5);
```

```
        return list;
    }

    @CacheEvict(cacheNames = "exampleCache", allEntries = true)
    public void evictCache() {
        log.info("delete all caches");
    }
}
```

- @Cacheable(cacheNames = “exampleCache”, key=“#useYn”) : 캐시를 사용할 메서드에 사용하는 어노테이션.
  - cacheNames는 ehcache.xml 에 설정한 cache alias와 동일해야 한다.
  - key는 캐시의 키로 사용되는 값을 spEL로 지정한다.
  - ehcache.xml에 설정한 exampleCache의 key-type, value-type이 일치해야 한다. 이 예제에서 key는 useYn이 String이므로 key-value는 java.lang.String이며 캐시되는 값은 SampleApiModel의 리스트이므로 value-type은 java.util.ArrayList이다. ArrayList에 저장되는 값은 SampleApiModel인데 이 클래스는 Serializable 인터페이스의 구현체이어야 한다.
  - 이 예제에서는 Thread.sleep() 메서드를 사용하여 DB 쿼리에 걸리는 시간을 시뮬레이트하여 처리.
- @CacheEvict(cacheNames = “exampleCache”, allEntries = true) : exampleCache에 저장된 캐시를 삭제하는 메서드에 사용하는 어노테이션

Repository 클래스 작성

- 이 예제에서는 DB를 사용하지 않으므로 생략.

Mapper XML 작성

- 이 예제에서는 DB를 사용하지 않으므로 생략.

Controller 클래스 작성

```
@Slf4j
@Api("OpenAPI for GSC Process Integration Project")
@RestController
public class SampleApiController {

    @Autowired SampleApiService sampleApiService;

    @GetMapping("/api/sample/return-map-api")
    @ApiOperation(value = "Map 반환 API", notes = "Map을 반환하는 API")
    public Map<String, Object> returnMapApi() {
        Map<String, Object> map = new HashMap<>();
```

```
        map.put("test1", 1);
        map.put("test2", 2);

        return map;
    }

    @GetMapping("/api/sample/cmm-code")
    @ApiOperation(value = "공통 코드 반환", notes = "공통 코드 전체 리스트 반환")
    public List<SampleApiModel> getAllCommonCode(@ApiParam(value = "사용 여
부", required = false, example = "Y") @RequestParam String useYn) {

        return sampleApiService.getAll(null);
    }

    @GetMapping("/api/sample/cache")
    @ApiOperation(value = "캐시 예제", notes = "EHCACHE 사용을 위한 예제")
    public List<SampleApiModel> getCache(@ApiParam(value = "사용 여부",
required = false, example = "Y") @RequestParam String useYn)  throws
InterruptedException {
        long start = System.currentTimeMillis();
        List<SampleApiModel> list = sampleApiService.getCache(useYn);
        long end = System.currentTimeMillis();

        log.info("쿼리 수행 시간 : {}ms", end-start);
        return list;
    }

    @GetMapping("/api/sample/evict-cache")
    @ApiOperation(value = "캐시 삭제", notes = "EHCACHE 캐시 삭제 예제")
    public void evictCache() {
        sampleApiService.evictCache();
        return;
    }
}
```

- getCache() : 캐시 사용 예제 서비스 호출 /api/sample/cache
- evictCache() : 캐시 삭제 예제 서비스 호출 /api/sample/evict-cache

References

기타 annotation

- @CachePut : 메서드 실행에 영향을 주지 않고 캐시를 갱신해야 하는 경우 사용
- @Caching : @CacheEvict이나 @CachePut을 여러개 지정해야 하는 경우에 사용
- @CacheConfig : 클래스 단위로 캐시설정을 동일하게 하는데 사용

Official EHCACHE site

- EHCACHE XML Configuration : https://www.ehcache.org/documentation/3.10/xml.html
- EHCACHE Official Main Site : https://www.ehcache.org/

- diskStore : 임시 저장 경로를 설정
  - path : 경로
- sizeOfPolicy : Cache에 저장할 사이즈 정책 설정
  - maxDepth : 최대값
  - maxDepthExceededBehavior : continue: 초과 된 최대 깊이에 대해 경고하지만 크기가 조정 된 요소를 계속 탐색 /abort: 순회를 중지하고 부분적으로 계산 된 크기를 즉시 반환

- defaultCache : 기본 캐시 설정 (Required)
  - eternal : "true" or "false"
  - timeToIdleSeconds :
  - timeToLiveSeconds :
  - overflowToDisk : "true" / "false"
  - diskPersistent : "true" / "false"
  - memoryStoreEvictionPolicy : "LRU"
- cache : 사용하고자 하는 캐시 설정
  - name: 코드에서 사용할 캐시 name (Required)
  - diskExpiryThreadIntervalSeconds:

maxEntriesLocalHeap 메모리에 생성 될 최대 캐시 갯수 0
maxEntriesLocalDisk 디스크에 생성 될 최대 캐시 갯수 0
eternal 영속성 캐시 설정 (지워지는 캐시인지?)
external = “true”이면, timeToIdleSecond, timeToLiveSeconds 설정이 무시됨
false
timeToIdleSecond 해당 초동안 캐시가 호출 되지 않으면 삭제 0
timeToLiveSeconds 해당 초가 지나면 캐시가 삭제 0
overflowToDisk 오버플로우 된 항목에 대해 disk에 저장할 지 여부 false
diskPersistent 캐시를 disk에 저장하여, 서버 로드 시 캐시를 말아 둘지 설정 false
diskExpiryThreadIntervalSeconds Disk Expiry 스레드의 작업 수행 간격 설정 0
memoryStoreEvictionPolicy 캐시의 객체 수가 maxEntriesLocalHeap에 도달하면, 객체를 추가하고 제거하는 정책 설정

- defaultCache는 반드시 구현해야 할 캐시 (직접 생성하는 캐시에 대한 기본 설정)
- cache는 하나의 캐시를 사용할 때마다 구현
- name 속성은 캐시의 이름을 지정하며, 코드에서는 이 캐시의 이름을 사용하여 사용할 Cache 인스턴스를 구한다.

@EnableCaching 설정
```java
@Configuration
```

```
@EnableCaching(proxyTargetClass = true, mode = AdviceMode.PROXY)
public class EHCacheConfig {
    @Bean
    public EhCacheManagerFactoryBean ehCacheManagerFactoryBean() {
        EhCacheManagerFactoryBean ehCacheManagerFactoryBean = new
EhCacheManagerFactoryBean();
        ehCacheManagerFactoryBean.setConfigLocation(new
ClassPathResource("config/ehcache.xml"));
        ehCacheManagerFactoryBean.setShared(true);
        return ehCacheManagerFactoryBean;
    }

    @Bean
    public EhCacheCacheManager ehCacheCacheManager(EhCacheManagerFactoryBean
ehCacheManagerFactoryBean) {
        EhCacheCacheManager ehCacheCacheManager = new EhCacheCacheManager();
ehCacheCacheManager.setCacheManager(ehCacheManagerFactoryBean.getObject());
        return ehCacheCacheManager;
    }
}

```

- @EnableCaching Annotation은 <cache:annotation-driven>와 마찬가지로 어노테이션 기반 캐시를 사용 할 수 있는 설정
- proxyTargetClass : class 기반 프록시를 생성함을 의미 (CGLIB라이브러리에 대한 의존성 필요)
- Mode : 어떤 Advisor 모듈을 선택할지에 대한 설정

- EHCache 설정방법 (Spring Framework)
- EHCache 설정방법 (Spring Boot)
- Cache에 대하여.. (Spring+EHCache)
- EHCache (1. 선택 및 특징)
- EHCache (2. 사용환경 셋팅)
- EHCache (3. 실제 사용)
- EHCache를 이용한 캐시 구현 2
- [Spring] ehCache2와 달라진 ehCache3 사용
- Springboot EhCache 3 - 환경설정부터 self-invocation 처리까지