http://www.egovframe.go.kr/wiki/doku.php?id=egovframework:rte2:fdl:ehcache

EhCache

Outline

In eGov Framework, the Cache Service assigns EhCache for guidance.

Contrary to CacheManager from EhCache used before Spring 3.1, you may now use CacheManager Abstraction for flexible cache structure. See the following for the description on EhCache and how to use EhCache in any edition before Spring 3.1:

Description

This Section describes how to configure and use EhCache:

Bootstrap Source

Refer to the following example for how to generate cache manager to use cache:

```
// Read the configuration file using the class path and generate the cache manager.
URL url = getClass().getResource("/ehcache-default.xml");
manager = new CacheManager(url);
```

The contents of /ehcache-default.xml read by way of getResource are as follows:

Configuration

```
<ehcache>
<diskStore path="user.dir/second"/>
    <cache name="sampleMem"
                maxElementsInMemory="3"
                eternal="false"
                timeToIdleSeconds="360"
                timeToLiveSeconds="1000"
                overflowToDisk="false"
                diskPersistent="false"
                memoryStoreEvictionPolicy="LRU">
    </cache>
</ehcache>
```

Basic Usage Source

Refer to the following example for how to obtain, read and write caches in cache manager as defined above:

```
// Obtain cache by cache Name
Cache cache = manager.getCache("sampleMem");

// 1. Input data in cache
cache.put(new Element("key1", "value1"));

// 2. Reading data out of cache input
Element value = cache.get("key1");

// 3. Deleting data in cache
cache.remove("key1");
```

- manager.getCache : Obtains cache using Manager
- cache.put : Puts data in cache (input other values for the same key value)
- cache.get : Reads data from cache
- cache.remove: Removes data from cache

Cache Algorithm

Cache works based upon the assigned volume of memory and removes unnecessary data when the storage exceeds the volume. An algorithm comes into play to determine necessity of required data, LRU, FIFO and LFU, described as follows:

LRU Algorithm

An algorithm intended to leave recently used caches, configured as follows:

Cofiguration

```
<cache name="sampleMem"
        maxElementsInMemory="3"
        ...
        memoryStoreEvictionPolicy="LRU">
```

Note that the maximum elements in memory (maxElementsInMemory) is configured for “3” in the foregoing example. See the following for how to confirm the result of the foregoing configuration:

Sample Source

```
Cache cache = manager.getCache("sampleMem");
...
cache.get("key2");
cache.get("key2");
cache.get("key1");
cache.get("key1");
cache.get("key3");

// 4. Put New element in cache.
cache.put(new Element("key4", "value4"));

// 5. get key2 but can't key2
assertNull("Can't get key2",cache.get("key2"));
```

FIFO Algorithm

A first in first out algorithm configured as follows:

Cofiguration

```
<cache name="sampleMemFIFO"
        maxElementsInMemory="3"
        ...
        memoryStoreEvictionPolicy="FIFO">
```

Note that the maximum elements in memory (maxElementsInMemory) is configured for “3” in the foregoing example. See the following for how to confirm the result of the foregoing configuration:

Sample Source

```
Cache cache = manager.getCache("sampleMemFIFO");
```

```
cache.put(new Element("key1", "value1"));
cache.put(new Element("key2", "value2"));
cache.put(new Element("key3", "value3"));
...
cache.get("key2");
cache.get("key2");
cache.get("key1");
cache.get("key1");
cache.get("key3");

// 4. Put New element in cache.
cache.put(new Element("key4", "value4"));

// 5. get key1 but can't key1
assertNull("Can't get key1",cache.get("key1"));
```

LFU Algorithm

An algorithm intended to remove the algorithm used least, configured as follows:

Cofiguration

```
<cache name="sampleMemLFU"
        maxElementsInMemory="3"
        ...
        memoryStoreEvictionPolicy="LFU">
```

Note that the maximum elements in memory (maxElementsInMemory) is configured for “3” in the foregoing example. See the following for how to confirm the result of the foregoing configuration:

Sample Source

```
Cache cache = manager.getCache("sampleMemLFU");
cache.put(new Element("key1", "value1"));
cache.put(new Element("key2", "value2"));
cache.put(new Element("key3", "value3"));
...
cache.get("key2");
cache.get("key2");
cache.get("key1");
cache.get("key1");
cache.get("key3");

// 4. Put New element in cache.
cache.put(new Element("key4", "value4"));

// 5. get key2 but can't key3
assertNull("Can't get key3",cache.get("key3"));
```

Cache Size & Device

In Cache Size & Decie you can configure the size of information and storage devices to be handled by cache.

Cache Device

By working cache device configurations you can configure disk management, transfer to disk in case of overflow and storage of file when flush is called. Refer to the following for how to configure the cache device:

Configuration

```
<cache name="sampleDisk"
        overflowToDisk="true"
        diskPersistent="true"
        ...>
</cache>
```

- overflowToDisk : Transfer to disk in case of overflow (true, false)
- diskPersistent : Storage of file in flush (true, false)

Refer to the following for how to configure the source codes:

Sample Source

```
Ehcache cache = manager.getCache("sampleDisk");

// 1. Put a content in Cache.
cache.put(new Element("key1", "value1"));

// 2. Get that item from Cache.
Element value = cache.get("key1");
assertEquals("value1", value.getValue().toString());

// 3. Flush the caches.
cache.flush();
File dataFile = new File(manager.getDiskStorePath()+ File.separator + "sampleDisk.data");
// 4. Save as a file.
assertTrue("File exists", dataFile.exists());
```

With the caches flushed as described in the above sample, you can also check the cache size to determine overflow to disk, as follows:

Cache Size

The maximum numbers of the objects in memory and on disk can be configured as follows:

Configuration

```
<cache name="sampleDisk"
        maxElementsInMemory="3"
        maxElementsOnDisk="2"
        overflowToDisk="true"
        ...>
</cache>
```

- maxElementsInMemory : The maximum number of elements in memory.
- maxElementsInMemory : The maximum number of elements on disk.

Refer to the following for how to configure the source codes:

Sample Source

```
Cache cache = manager.getCache("sampleDisk");

// 1. Put 3 contents in Cache.
cache.put(new Element("key1", "value1"));
cache.put(new Element("key2", "value2"));
cache.put(new Element("key3", "value3"));
```

```
// 2. Put Fourth content in Cache.
cache.put(new Element("key4", "value4"));
assertEquals(3, cache.getMemoryStoreSize()); // A total of three caches are to be maintained in a memory.
assertEquals(1, cache.getDiskStoreSize());      // Transfer of a cache in memory.

// 3. Put 5~7 contens in Cache.
cache.put(new Element("key5", "value5"));
cache.put(new Element("key6", "value6"));
cache.put(new Element("key7", "value7"));

// Transfer of caches regardless of the Disk Max Size.
assertEquals(3, cache.getMemoryStoreSize()); // A total of three caches are to be maintained in a memory.
assertEquals(4, cache.getDiskStoreSize());      // A total of 4 caches are to be maintained in the disk.

// A total of 2 caches in memory are transferred to the disk.
  cache.flush();
// Change of value as per the max size of the disk.
assertEquals(0, cache.getMemoryStoreSize()); // No caches available in the memory.
assertEquals(2, cache.getDiskStoreSize());      // 2 caches are remaining on the Disk as per DiskMaxSize.
```

In the foregoing example the memory is out to the disk (Disk) regardless of how maximum size (MaxSize) is configured and this only is the case for cache.put. You can also find the memory is capped at the maximum number of disk when flushed (Flush).

Distributed Cache

Like how Distributed Cache works, Ehcache supports RMI, JGROUP and JMS. In this Section, we're going to describe how JMS support is configured using JGROUP and ActiveMQ. See EhcacheUserGuide for more information.

Using JGroups

JGroups allows users to generate groups and send/receive messages within the same group, using the multicast-based communication toolkit. Visit Website for more information.

Configuration

```
    <cacheManagerPeerProviderFactory
class="net.sf.ehcache.distribution.jgroups.JGroupsCacheManagerPeerProviderFactory"

properties="connect=UDP(mcast_addr=224.10.10.10;mcast_port=5555;ip_ttl=32;
                                        mcast_send_buf_size=150000;mcast_recv_buf_size=80000):
                                        PING(timeout=2000;num_initial_members=6):
                                        MERGE2(min_interval=5000;max_interval=10000):
                                        FD_SOCK:VERIFY_SUSPECT(timeout=1500):
                                        pbcast.NAKACK(gc_lag=10;retransmit_timeout=3000):
                                        UNICAST(timeout=5000):
                                        pbcast.STABLE(desired_avg_gossip=20000):
                                        FRAG:

pbcast.GMS(join_timeout=5000;join_retry_timeout=2000;shun=false;print_local_addr=false)"
                                        propertySeparator="::"/>
    <cache name="cacheSync"
            maxElementsInMemory="1000"
            eternal="false"
            timeToIdleSeconds="1000"
            timeToLiveSeconds="1000"
            overflowToDisk="false">
        <cacheEventListenerFactory class="net.sf.ehcache.distribution.jgroups.JGroupsCacheReplicatorFactory"
```

```
                                        properties="replicateAsynchronously=false, replicatePuts=true,
                                                                                                        re
                                                                                                        re
    </cache>
```

Refer to the following for how to configure the source codes:

Sample Source

```
    // Read the information configured in the aforementioned configuration file
    URL url = this.getClass().getResource("/ehcache-distributed-jgroups.xml");
    // Generate a couple Cache Managers
    manager1 = new CacheManager(url);
    manager2 = new CacheManager(url);

    for (int i = 0; i < 10 ; i++) {
        manager1.getEhcache(CACHE_SYNC).put(new Element(new Integer(i), "value"));
    }
    // Replication latency
    Thread.currentThread().sleep(100);
    // Check out the replication is made to Manager 2
    assertTrue(manager1.getEhcache(CACHE_SYNC).getKeys().size() ==
manager2.getEhcache(CACHE_SYNC).getKeys().size() );
```

Using ActiveMQ

ActiveMQ is an open source messaging solution for JMS. Visit Website for more information.

Configuration

```
    <cacheManagerPeerProviderFactory
            class="net.sf.ehcache.distribution.jms.JMSCacheManagerPeerProviderFactory"

properties="initialContextFactoryName=egovframework.rte.fdl.cache.distribute.TestActiveMQInitialContextFactory,
                providerURL=tcp://localhost:61616,
                replicationTopicConnectionFactoryBindingName=topicConnectionFactory,
                getQueueConnectionFactoryBindingName=queueConnectionFactory,
                replicationTopicBindingName=ehcache,
                getQueueBindingName=ehcacheGetQueue"
            propertySeparator=","
            />

    <cache name="CacheAsync"
            maxElementsInMemory="1000"
            eternal="false"
            timeToIdleSeconds="1000"
            timeToLiveSeconds="1000"
            overflowToDisk="false">

            <cacheEventListenerFactory class="net.sf.ehcache.distribution.jms.JMSCacheReplicatorFactory"
                                        properties="replicateAsynchronously=true,
                                                    replicatePuts=true,
                                                    replicateUpdates=true,
                                                    replicateUpdatesViaCopy=true,
                                                    replicateRemovals=true,
                                                    asynchronousReplicationIntervalMillis=1000"
                                         propertySeparator=","/>
    </cache>
```

Refer to the following for how to configure the source codes:

Sample Source

```
URL url = this.getClass().getResource("/ehcache-distributed-activemq.xml");
manager1 = new CacheManager(url);
manager2 = new CacheManager(url);

cacheName = "CacheAsync";
for (int i = 0; i < 10 ; i++) {
    manager1.getEhcache("CacheAsync").put(new Element(new Integer(i), "value"));
}

// Replication latency
Thread.currentThread().sleep(1000);

assertTrue(manager1.getEhcache("CacheAsync").getKeys().size() ==
manager2.getEhcache("CacheAsync").getKeys().size() );
```

6. Spring Integration

In this Section we'll describe how Ehcache works in the older version of Spring 3.1 and how yo ucan capitalize on the cache services working the configurations.

Configuration - Spring Application Context

```
<bean id="ehcache" class="org.springframework.cache.ehcache.EhCacheFactoryBean">
    <property name="cacheManager">
        <bean class="org.springframework.cache.ehcache.EhCacheManagerFactoryBean">
            <property name="configLocation" value="classpath:ehcache-default.xml"/>
        </bean>
    </property>
</bean>
```

These are how you can get all those caches without having to reach out to the Manager. Note that ehcache-default.xml defined in ConfigLocation is the same configuration file with the Configuration in Cache Basic.

Sample Source

```
@Resource(name="ehcache")
Ehcache gCache ;

// cache Name 을 가지고 cache 찾기
  Ehcache cache = gCache.getCacheManager().getCache("sampleMem");

cache.put(new Element("key1", "value1"));
Element value = cache.get("key1");
```

In the foregoing example you may have learned how Ehcache is loaded and read by way of getCacheManager(). You can follow the descriptions above for how the rest of functions work.

References

EhCache