

# Anyframe Cache Plugin



Version 1.0.2

저작권 © 2007-2011 삼성SDS

본 문서의 저작권은 삼성SDS에 있으며 Anyframe 오픈소스 커뮤니티 활동의 목적하에서 자유로운 이용이 가능합니다. 본 문서를 복제, 배포할 경우에는 저작권자를 명시하여 주시기 바라며 본 문서를 변경하실 경우에는 원문과 변경된 내용을 표시하여 주시기 바랍니다. 원문과 변경된 문서에 대한 상업적 용도의 활용은 허용되지 않습니다. 본 문서에 오류가 있다고 판단될 경우 이슈로 등록해 주시면 적절한 조치를 취하도록 하겠습니다.

---

I. Introduction .....	1
II. Cache .....	2
1. Configuration .....	3
2. Samples .....	5
3. Resources .....	7

---

# I.Introduction

cache plugin은 opensymphony의 OSCache [<http://www.opensymphony.com/oscache/>]를 간편하게 활용할 수 있도록 하기 위해 제공되는 Cache Service의 기본 활용 방법을 가이드하기 위한 샘플 코드와 이를 활용하는데 필요한 참조 라이브러리들로 구성되어 있다.

## Installation

Command 창에서 다음과 같이 명령어를 입력하여 cache plugin을 설치한다.

```
mvn anyframe:install -Dname=cache
```

installed(mvn anyframe:installed) 혹은 jetty:run(mvn clean jetty:run) command를 이용하여 설치 결과를 확인해볼 수 있다.

## Dependent Plugins

Plugin Name	Version Range
<a href="http://dev.anyframejava.org/docs/anyframe/plugin/optional/query/1.1.2/reference/htmlsingle/query.html">query [http://dev.anyframejava.org/docs/anyframe/plugin/optional/query/1.1.2/reference/htmlsingle/query.html]</a>	2.0.0 > *

---

## II.Cache

Cache Service는 opensymphony의 OSCache [<http://www.opensymphony.com/oscache/>]를 기반으로 개발되었으며, 상태를 공유할 수 있는 객체를 cache하는 기능을 제공하는 서비스이다. 변경이 자주 일어나지 않지만 사용빈도가 높고 생성하는데 비용이 많이 드는 객체일 경우, Cache를 이용하면 다음과 같은 장점을 얻을 수 있다.

- 자주 접근하는 데이터를 매번 데이터베이스로부터 fetch할 필요가 없으므로 오버헤드가 줄어든다.
  - 객체를 매번 생성하지 않기 때문에 메모리를 효율적으로 사용할 수 있다.
-

# 1. Configuration

CacheServiceImpl은 OSCache의 속성을 그대로 따라, Caching된 객체의 상태를 보호하기 위해 ThreadSafe한 Cache를 제공한다. CacheServiceImpl에서 OSCache가 제공하는 Cache를 사용하기 위해서는 다음과 같은 설정을 필요로 한다.

Property Name	Description	Required	Default Value
cache	사용할 Cache의 속성을 정의한 Cache의 id를 참조한다.	Y	N/A

다음은 OSCache가 가지는 설정 정보에 대한 설명이다.

Property Name	Description	Required	Default Value
cache.memory	메모리 Cache를 사용할 것인지 정의한다. false로 설정되면 메모리로 캐싱될 수 없다.	N	true
cache.capacity	Cache에 저장할 수 있는 object의 최대 갯수를 지정한다. 음수로 설정되면 이 기능을 사용하지 않는다. 캐싱 가능한 object의 갯수를 제한하지 않는다.	N	-1
cache.algorithm	caching algorithm의 classname을 지정한다. 이 클래스는 com.opensymphony.oscache.base.algorithm.AbstractConcurrentReadCache를 extend 해야한다. cache capacity가 양수로 설정되면 default algorithm으로 LRUCache가 사용되고, 음수로 설정되면 com.opensymphony.oscache.base.algorithm.UnlimitedCache가 사용된다.	N	N/A
cache.unlimited.disk	Persistence cache의 size를 제한할 것인지 또는 in-memory cache와 동일한 사이즈로 제한할 것인지를 나타낸다. 이 값이 true로 설정되면 persistent cache는 제한없이 사용될 수 있다.	N	false
cache.blocking	새로운 content를 캐싱하거나 이미 캐싱된 content를 검색할 때 block waiting 해야 하는지를 정의한다.	N	false
cache.persistence.class	Persistence cache를 사용하고자 할 때 Persistence cache를 구현한 classname을 정의한다. 이 클래스는 PersistenceListener를 extend 해야한다.	N	N/A
cache.persistence.overflow	Overflow mode일 때 Persistence Cache를 사용할지 지정한다.	N	false
cache.event.listeners	Cache에 적용한 event handler를 지정한다. event handler가 여러개 일 경우 각각의 classname을 콤마로 구분하여 정의한다.	N	N/A
cache.cluster.properties	JavaGroupsBroadcastingListener를 사용할 때 이 property를 정의한다. JavaGroups channel properties를 사용한다. JavaGroups의 실행을 제어할 수 있다.	N	
cache.cluster.multicast	JavaGroupsBroadcastingListener를 사용할 때 이 property를 정의한다. broadcasting을 사용하기 위해 JavaGroups는 multicast IP를 사용해야 한다.	N	231.12.21.132
cache.cluster.jms.notification	JMSBroadcastingListener 또는 JMSBroadcastingListener를 사용할 때 이 property를 정의한다. JMS connection factory를 사용한다.	Y	N/A

Property Name	Description	Required	Default Value
cache.cluster.jms.topic	JMSBroadcastingListener 또는 JMSBroadcastingListener를 사용할때 이 property를 정의한다. 이것은 JMS topic name이다.	Y	N/A
cache.cluster.jms.topic	JMSBroadcastingListener 또는 JMSBroadcastingListener를 사용할때 이 property를 정의한다. 이 노드의 이름은 cluster에 존재하고, 각각의 node마다 unique한 값을 갖는다.	Y	N/A
cache.path	DiskPersistenceListener를 사용할 때 이 property를 정의한다. 데이터를 캐싱하기 위한 path를 지정한다.	Y	N/A
cache.persistence.disk.algorithm	DiskPersistenceListener로 간단한 cache key를 생성하기 위한 hash algorithm이다.	N	MD5

---

## 2.Samples

다음은 CacheServiceImpl의 속성 설정 및 테스트 코드에 대한 예제이다.

- **Configuration**

다음은 CacheServiceImpl과 사용할 Cache의 속성을 정의한 context-cache.xml 의 일부이다. 아래 속성 정의 파일 내용에 따르면, cacheAdministrator Bean에 테스트에서 사용할 Cache의 속성이 정의되어 있다. 또한 cacheService Bean에서 cacheAdministrator의 getCache 메소드를 통해 도출한 캐시를 참조하고 있다.

```
<bean id="cacheService" class="org.anyframe.cache.impl.CacheServiceImpl">
  <property name="cache" ref="cache" />
</bean>

<bean id="cache" factory-bean="cacheAdministrator" factory-method="getCache" />

<bean id="cacheAdministrator"
  class="com.opensymphony.oscache.general.GeneralCacheAdministrator"
  destroy-method="destroy">
  <constructor-arg index="0">
    <props>
      <prop key="cache.capacity">10</prop>
    </props>
  </constructor-arg>
</bean>
```

- **TestCase**

다음은 앞서 정의한 속성 설정을 기반으로 하여 Cache 내에 특정 데이터를 저장하고 추출하는 테스트 케이스 코드의 일부이다.

```
public void manageGenre() throws Exception {
  // 1. create a new genre
  Genre genre = new Genre();
  genre.setName("Western");
  genreService.create(genre);
  putInCache(genre);

  // 2. assert - create
  genre = getGenreByPk(genre.getGenreId());
  assertNotNull("fail to fetch a genre", genre);
  assertEquals("fail to compare a genre name", "Western", genre.getName());

  // 3. update a name of genre
  String name = "Western " + System.currentTimeMillis();
  genre.setName(name);
  genreService.update(genre);
  putInCache(genre);

  // 4. assert - update
  genre = getGenreByPk(genre.getGenreId());
  assertNotNull("fail to fetch a genre", genre);
  assertEquals("fail to compare a updated name", name, genre.getName());

  // 5. remove a genre
  genreService.remove(genre.getGenreId());
}

private Genre getGenreByPk(String genreId) throws Exception{
```

```
Map<String, Genre> resultMap = (HashMap<String, Genre>) cacheService
    .getFromCache("genreList");
Genre genre = resultMap.get(genreId);
return genre;
}
```

---

## 3.Resources

- 참고자료
  - OSCache Home [<http://www.opensymphony.com/oscache/>]
  - OS Cache Configuration Guide [<http://www.opensymphony.com/oscache/wiki/Configuration.html>]