1-) Configuração

1.1-) Adicionar as duas dependências abaixo no pom.xml do projeto EJB

<dependency>

<groupId>net.sf.ehcache</groupId>

<artifactId>ehcache-core</artifactId>

<version>2.6.6</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-ehcache</artifactId>

<version>4.1.0.Final</version>

</dependency>

1.2-) Adicionar as seguintes propriedades no persistence.xml

<!-- 2nd level cache -->

<property name=*"hibernate.cache.use\_second\_level\_cache"* value=*"true"* />

<property name=*"hibernate.cache.use\_query\_cache"* value=*"false"* />

<property name=*"hibernate.cache.provider\_class"* value=*"org.hibernate.cache.ehcache.SingletonEhCacheRegionFactory"* />

<property name=*"hibernate.cache.region.factory\_class"* value=*"org.hibernate.cache.ehcache.EhCacheRegionFactory"*/>

<property name=*"net.sf.ehcache.configurationResourceName"* value=*"/ehcache.xml"*/>

<property name=*"hibernate.generate\_statistics"* value=*"true"* />

<property name=*"hibernate.cache.region\_prefix"* value=*""* />

</properties>

<shared-cache-mode>ENABLE\_SELECTIVE</shared-cache-mode>

Onde:

1.2.1-) <property name=*"hibernate.cache.use\_second\_level\_cache"* value=*"true"* />

Propriedade para ativar o cache de segundo nível

1.2.2-) <property name=*"hibernate.cache.use\_query\_cache"* value=*"false"* />

Propriedade para ativar o cache de queries. (Inicialmente desabilitado esse tipo de cache. Será explicado posteriormente)

1.2.3-) <property name=*"hibernate.cache.provider\_class"* value=*"org.hibernate.cache.ehcache.SingletonEhCacheRegionFactory"* />

Classe gerenciadora do CacheManager. Como possuímos apenas uma session factory, estamos utilizando o *SingletonEhCacheRegionFactory*

1.2.4-) <property name=*"net.sf.ehcache.configurationResourceName"* value=*"/ehcache.xml"*/>

Local do arquivo de configuração do ehcache.

1.2.5-) <property name=*"hibernate.generate\_statistics"* value=*"true"* />

Propriedade que ativa a geração de estatísticas dos itens em cachê.

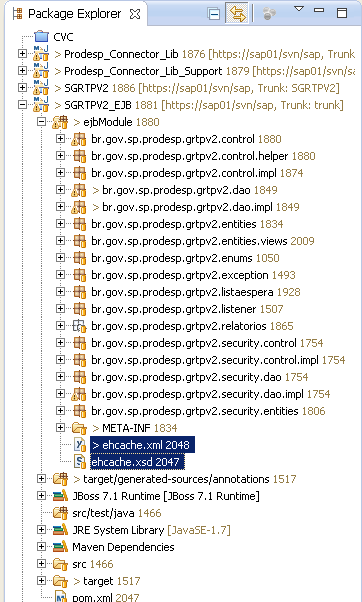
1.2.6-) <property name=*"hibernate.cache.region\_prefix"* value=*""* />

Propriedade que indica se os caches definidos devem possuir um prefixo. Está em branco pois assim geramos apenas com o nome das entidades. Ex.: <cache name=*"br.gov.sp.prodesp.grtpv2.entities.OutrosSSP" ...*

1.2.7-) <shared-cache-mode>ENABLE\_SELECTIVE</shared-cache-mode>

Configuração que ativa o cache apenas para entidades que possuem @Cacheable(true) declarado.

1.3-) Criados arquivos ehcache.xml e ehcache.xsd na seguinte estrutura(esses arquivos podem ser baixados do site: <http://ehcache.org/>



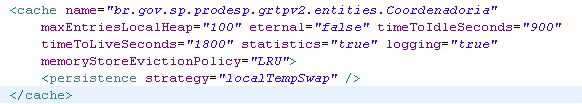
2-) Utilização

Para adicionar uma nova entidade no cache, são necessários apenas dois passos:

2.1-) Marcar a entidade**(Sempre implementando Serializable)** desejada como @Cacheable(true):



2.2-) Criar um cache específico para a entidade no arquivo ehcache.xml:



3-) ehcache.xml

Seguem detalhes das propriedades do arquivo de configuração do ehcache.

3.1-) Tag <ehcache>

<ehcache xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:noNamespaceSchemaLocation=*"ehcache.xsd"* updateCheck=*"true"*

monitoring=*"autodetect"* dynamicConfig=*"true"* name=*"ehcache.grtp"* maxBytesLocalDisk=*"1G"*>

Aqui vale ressaltar a propriedade maxBytesLocalDisk=*"1G"* que criará um arquivo no servidor para realizar swap em disco caso a capacidade de memória de determinado cache ultrapasse o limite definido.

3.2-) Tag <cache>

<cache name=*"br.gov.sp.prodesp.grtpv2.entities.Coordenadoria"*

maxEntriesLocalHeap=*"100"* eternal=*"false"* timeToIdleSeconds=*"900"*

timeToLiveSeconds=*"1800"* statistics=*"true"* logging=*"true"*

memoryStoreEvictionPolicy=*"LRU"*>

<persistence strategy=*"localTempSwap"* />

</cache>

3.2.1-) name=*"br.gov.sp.prodesp.grtpv2.entities.Coordenadoria"*

Define o nome do cache. No nosso caso, como estamos aplicando cache de segundo nível, a entidade(com pacote) indica o cache que a mesma pertence.

3.2.2-) maxEntriesLocalHeap=*"100"*

Define a quantidade de objetos em memória. Quando essa quantidade é ultrapassada, se definida a utilização de swap em disco, os dados são adicionados em um arquivo no servidor. Se não definida a utilização de swap em disco, utiliza a propriedade memoryStoreEvictionPolicy=*"LRU"(Será falada mais adiante)*

3.2.3-) eternal=*"false"*

Define que os itens em cache não permanecerão eternamente em cache.

3.2.4-) timeToIdleSeconds=*"900"*

Define o tempo(em segundos) que um dado pode ficar no cache sem nenhum acesso. Ou seja, nesse exemplo, se um dado ficar 15 minutos sem nenhum acesso, o mesmo sairá do cache.

3.2.5-) timeToLiveSeconds=*"1800"*

Define o tempo(em segundos) que um dado pode ficar no cache independente de uso.

3.2.6-) statistics=*"true"* logging=*"true"*

Permite a geração de estatísticas e log para o cache em questão.

3.2.7-) memoryStoreEvictionPolicy=*"LRU"*

Define a lógica que será aplicada quando o limite de armazenamento em cache é ultrapassada. No sistema é utilizado para todos os caches a política LRU - least recently used.

3.2.8-) <persistence strategy=*"localTempSwap"* />

Ativa o uso de arquivos em disco. Esses arquivos não são persistidos e a qualquer restart do servidor de aplicação, são apagados.

3.3-) Tag <diskStore>

Identifica onde os arquivos que o cache utilizará(Se configurado) serão gerados.

4-) Query Cache

Utilizado para evitar que uma query já executada anteriormente acesse novamente o banco de dados. Isso ajuda bastante principalmente quando o banco é remoto. Porém deve-se tomar muito cuidado ao escolher uma query para adicionar ao cache, pois a string da query e os respectivos parâmetros ficarão alocados em memória.

Quando utilizar:

- Quando uma mesma query é usada diversas vezes pela aplicação. Um exemplo de utilização no GRTP é a lista de Cadeias e Distritos. Todas as telas de pesquisa carregam essas informações. Portanto acessaremos o banco apenas uma vez.

Quando não utilizar:

- Quando uma query recebe muitos parâmetros e esses parâmetros tem muita variação. Cada pesquisa com parâmetros diferentes é tratada como uma pesquisa nova para o cache. Ou seja:

SELECT \* FROM CARGO WHERE ID = 1 AND STATUS = ‘A’

SELECT \* FROM CARGO WHERE ID = 2 AND STATUS = ‘A’

SELECT \* FROM CARGO WHERE ID = 1 AND STATUS = ‘I’

SELECT \* FROM CARGO WHERE ID = 2 AND STATUS = ‘I’

No exemplo acima serão adicionadas 4 queries em cache.

Ao utilizar query cache, obrigatoriamente deve-se utilizar Second Level Cache.

Para utilizar o cache de query, é necessário apenas setar a seguinte propriedade no persistence.xml:

<property name=*"hibernate.cache.use\_query\_cache"* value=*"true"* />

E para cada query que será adicionada ao cache, adicionar a seguinte linha:

Javax.persistence.Query.setHint("org.hibernate.cacheable", true);

**Referências:**

Strategy Options: <http://ehcache.org/documentation/configuration/fast-restart>

Eviction Policy: <http://ehcache.org/apidocs/net/sf/ehcache/store/MemoryStoreEvictionPolicy.html#LRU>

Cache Configuration(General): <http://ehcache.org/documentation/2.5/configuration/configuration#dynamically-changing-cache-configuration>

SharedCacheMode(persistence.xml): <http://docs.oracle.com/javaee/6/api/javax/persistence/SharedCacheMode.html>

SingletonEhCacheRegionFactory : <http://ehcache.org/apidocs/index.html?net/sf/ehcache/hibernate/SingletonEhCacheRegionFactory.html>

Ehcache: <http://ehcache.org/>

\*\*Sites que auxiliaram o desenvolvimento

Query Cache(Problems):

<http://tech.puredanger.com/2009/07/10/hibernate-query-cache/>

<http://darren.oldag.net/2009/05/hibernate-wars-query-cache-strikes-back.html>

<http://darren.oldag.net/2008/11/hibernate-query-cache-dirty-little_04.html>

JPA/Hibernate Cache:

<http://acupof.blogspot.com.br/2008/01/background-hibernate-comes-with-three.html>

<http://www.javalobby.org/java/forums/t48846.html>

<http://docs.jboss.org/hibernate/core/4.0/manual/en-US/html/performance.html#performance-cache>

<http://richardchesterwood.blogspot.com.br/2013/01/configuring-cacheconcurrencystrategy-in.html>

<http://www.gridshore.nl/2008/04/29/using-ehcache-and-verifying-that-it-works-with-jpa-and-springframework/>

<http://javathoughts.capesugarbird.com/2009/02/jpa-ejb3-persistence-and-ehcache.html>