Prog. Orientada a Objetos e Mapeamento Objeto-Relacional – IMD0104

Cache

João Carlos Xavier Júnior

jcxavier@imd.ufrn.br

INSTITUTO METRÓPOLE DIGITAL

UFRN UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Cache de Primeiro Nível

- Provido pelo **framework de ORM**.
- Cache de **primeiro nível**, também chamado de **L1** ou simplesmente de cache.
- Todas **as requisições devem passar** por ele.
- **Previne** que um mesmo **objeto seja carregado duas vezes**, reduzindo assim o consumo de memória e solicitações ao banco de dados.
- Também é chamado de **cache de transação**.

Cache de Segundo Nível

- A estratégia de concorrência é um **mediador** entre o **cache** e o **solicitante**.
- Existem 4 níveis de concorrência:
  - **Transactional**: garante o isolamento da transação total até ***reptable read***. É usada para dados que são mais lidos que atualizados.
  - **Read-write**: mantém o nível de isolamento ***read-commited***.
  - **Nonstrict-read-write**: **não garante consistência** entre o **cache** e o **banco**.
  - **Read-only**: dados que nunca mudam.

Concorrência

- **Exemplo:**

```
@Entity
@Table(name = "estado")
@Cache(usage = CacheConcurrencyStrategy.NONSTRICT_READ_WRITE)
public class Estado implements Serializable {

    private static final long serialVersionUID = 1L;

    private Long codigo;
    private String nome;
    private List<Cidade> cidades;

    // getters e setters

}
```

Formas de Cache de 2º Nível

- **Formas:**
  - Utilizando o padrão do **JPA**.
  - Utilizando o padrão do **Hibernate**.

Configurando o cache no JPA

- Uso de **Cache no JPA 2.0** é fácil de configurar.
- Formas de configurar:
  - Forma global através da **unidade de persistência**; ou
  - Forma mais específica **classe por classe**.
- A configuração do cache na unidade de persistência é determinada através do elemento **shared-cached-mode** no arquivo **persistence.xml**.

Definição do modo de Cache

Modo	Descrição
ALL	Todas as entidades são armazenadas no cache de segundo nível.
NONE	Nenhum entidade é armazenada no cache.
ENABLE_SELECTIVE	Habilita o Cache para as entidades que tiverem a anotação @Cacheable
DISABLE_SELECTIVE	Habilita o cache para todas as entidade, menos as que tem @Cacheable(false)
UNSPECIFIED	O comportamento do cache não é especificado, será baseado no cache provider.

Configurando o cache no JPA

```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0"
    xmlns="http://java.sun.com/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
                        http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">

    <persistence-unit name="ConexaoDB" transaction-type="RESOURCE_LOCAL">
        <provider>org.hibernate.ejb.HibernatePersistence</provider>
        <class>br.edu.unirn.model.Grupo</class>
        <class>br.edu.unirn.model.Contato</class>
        <shared-cache-mode>ENABLE_SELECTIVE</shared-cache-mode>

        <properties>

        </properties>
    </persistence-unit>
</persistence>
```

Configurando o cache no JPA

- Configurando dentro das classes:
  - Uso da propriedade **javax.persistence.sharedCache.mode** passada como parâmetro para o **EntityManagerFactory** em tempo de execução.

```
public static EntityManager getInstance(){
    if(em == null){
        EntityManagerFactor emf =
                Persistence.createEntityManagerFactory("ConexaoDB", new Properties()
                .add("javax.persistence.sharedCache.mode","ENABLE_SELECTIVE"));
        em = emf.createEntityManager();
    }
    return em;
}
```

Configurando o cache no JPA

- Configurando dentro das classes:
  - Utilizando a anotação (**@Cacheable**).

```
@Entity
@Cacheable (true) // @Cacheable
public class Autor implements Serializable{
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private Integer id;
    private String nome;
    private String cpf;

    @ManyToMany
    @JoinTable(name="autor_livro",
    joinColumns = @JoinColumn(name="id_autor"),
    inverseJoinColumns = @JoinColumn(name="id_livro"))
    private Collection<Livro> livros;

    // Getters and Setters
```

Gerenciando o Cache dinamicamente

- É possível gerenciar o **cache dinamicamente** em tempo de execução.
- Reescrever alguns comportamentos das operações de leitura e escrita:
  - **CacheRetrieveMode**
  - **CacheStoreMode**

Gerenciando o Cache dinamicamente

- O modo de **leitura** tem duas opções:
  - **CacheRetrieveMode.USE**: para se utilizar os dados diretamente do cache.
  - **CacheRetrieveMode.BYPASS**: para não acessar o cache, obter direto do banco de dados.
- O modo de **gravação** tem três opções:
  - **CacheStoreMode.USE**: um objeto é inserido no cache depois de ter sido recuperado ou comitado no banco de dados.
  - **CacheStoreMode.BYPASS**: para não inserir um objeto no cache.

Gerenciando o Cache dinamicamente

- O modo de gravação tem três opções:
  - **CacheStoreMode.REFRESH**: usado para atualizar um objeto no cache, caso exista.

Gerenciando o Cache dinamicamente

- **Exemplo:**

```
public List<Autor> buscaPorCPF(String cpf){
    TypedQuery<Autor> q =em.createQuery(
            "SELECT a.nome FROM Autor a WHERE a.cpf = : cpf", Autor.class);
    q.setProperty("javax.persistence.cache.retrieveMode",
            CacheRetrieveMode.BYPASS);
    q.setProperty("javax.persistence.cache.storeMode",
            CacheStoreMode.REFRESH);
    q.setParameter("cpf", cpf);
    return q.getResultList();
}
```

Cache no Hibernate

- Através do **Hibernate** é possível especificar qual o **provedor de cache** que será utilizado.
- Ou seja, qual a solução responsável por **coordenar** as informações no cache.
- Provedores de cache (Cache Provider):
  - **EHCache** (utilizado como padrão);
  - OSCache;
  - SwarmCache;
  - Jboss Cache.

Configuração EHCache

- persistence.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0"
    xmlns="http://java.sun.com/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
                        http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
    <persistence-unit name="ConexaoDB" transaction-type="RESOURCE_LOCAL">
        <provider>org.hibernate.ejb.HibernatePersistence</provider>
        <class>br.edu.unirn.model.Grupo</class>
        <class>br.edu.unirn.model.Contato</class>
        <shared-cache-mode>NONE</shared-cache-mode>
        <properties>
            <property name="hibernate.cache.region.factory_class"
                value="org.hibernate.cache.EhCacheRegionFactory" />
            <property name="hibernate.cache.provider_class"
                value="org.hibernate.cache.EhCacheProvider" />
            <property name="hibernate.cache.use_query_cache"
                value="true" />
            <property name="hibernate.cache.use_second_level_cache"
                value="true" />
            <property name="hibernate.generate_statistics"
                value="true" />
        </properties>
    </persistence-unit>
</persistence>
```

Configuração EHCache

- **ehcache.xml:**

```
<?xml version="1.0" encoding="UTF-8"?>
<ehcache>
    <diskStore path="java.io.tmpdir" />

    <cache name="br.edu.unirn.model.Contato" maxElementsInMemory="2000"
        eternal="false" timeToIdleSeconds="1800" timeToLiveSeconds="3600"
        overflowToDisk="false" />

    <defaultCache maxElementsInMemory="10000" eternal="false"
        timeToIdleSeconds="120" timeToLiveSeconds="120" overflowToDisk="true"
        diskSpoolBufferSizeMB="30" maxElementsOnDisk="10000000"
        diskPersistent="false" diskExpiryThreadIntervalSeconds="120"
        memoryStoreEvictionPolicy="LRU" />
</ehcache>
```

Configuração EHCache

- **diskStore**: local aonde será armazenado o cache.
- **cache** (name): nome da entidade que fará uso do cache.
- **defaultCache**: configuração de cache geral. Serve para todas as classe que não possuem cache específico configurado.

Dúvidas...