DobraDbServiceManager.java

- Substituicao de:

import org.hibernate.Criteria

- por jakarta:

import jakarta.persistence.criteria.CriteriaBuilder;
import jakarta.persistence.criteria.CriteriaQuery;
import jakarta.persistence.criteria.Predicate;
import jakarta.persistence.criteria.Root;

DobreDbInternalService.java

- Substituicao de:

import org.hibernate.Criteria
import org.hibernate.criterion.Example;
import org.hibernate.criterion.Restrictions;

- por jakarta:

import jakarta.persistence.criteria.CriteriaBuilder;
import jakarta.persistence.criteria.CriteriaQuery;
import jakarta.persistence.criteria.Predicate;
import jakarta.persistence.criteria.Root;

1. Remoção da API org.hibernate.Criteria e QBE
   A antiga API de Criteria (org.hibernate.Criteria, Example, Restrictions) foi removida no Hibernate 6 em favor da JPA Criteria ou de HQL direto. Substituímos a busca dinâmica por HQL, consultando diretamente o domainKey de cada entidade.
   Stack Overflow
   Thorben Janssen

2. Uso de setParameter em vez de setString
   O método Query.setString(...) foi removido no Hibernate 6. Agora devemos usar Query.setParameter(...) para parametrizar nossas queries.

3. Substituição de createSQLQuery por createNativeQuery
   Em Hibernate 6, Session.createSQLQuery(...) está depreciado/removido. Utilizamos Session.createNativeQuery(...) para executar SQL nativo.
   JBoss Dokumentation

4. Obtenção de esquema via getSessionFactoryOptions()
   O método getSettings() de SessionFactoryImplementor foi removido a partir do Hibernate 6. Passamos a usar getSessionFactoryOptions().getDefaultSchemaName() para recuperar o esquema padrão.
   javadoc.io

5. Adequação do método getObject(...)
   Ajustamos a criação de Query tipada (Query) e usamos uniqueResult() diretamente, evitando cast genérico e garantindo tipagem segura.

Alteração	Justificativa
`SessionImplementor` → `SharedSessionContractImplementor`	O `SessionImplementor` foi refatorado para `SharedSessionContractImplementor` no Hibernate 6.
`setFlushMode(FlushMode.COMMIT)`	O `FlushMode` do Hibernate foi migrado, e agora há distinção clara entre o JPA (`jakarta.persistence.FlushModeType`) e o Hibernate (`org.hibernate.FlushMode`). Ambos foram aplicados.
`Metamodel.entity(String)` removido	A API de metamodelo de JPA não aceita mais `String` como parâmetro, requer `Class<T>`. Uma função auxiliar `resolveEntityClass` foi adicionada.
`createCriteria` (comentado)	A API `createCriteria` foi descontinuada no Hibernate 6.1; o uso correto é via JPA CriteriaBuilder.
Substituição de imports e métodos descontinuados	Todas as referências obsoletas foram atualizadas conforme documentação oficial Hibernate 6.1.7.

DobreDbServiceManager

1. FlushModeType

Antes: session.setFlushMode(FlushMode.COMMIT) usava o enum de Hibernate (org.hibernate.FlushMode).

Depois: session.setFlushMode(FlushModeType.COMMIT) passa a usar o enum JPA (jakarta.persistence.FlushModeType), pois o método em Hibernate 6 espera FlushModeType.

2. Critérios (Criteria API)

O antigo org.hibernate.Criteria e Restrictions foram removidos em Hibernate 6.

Criamos um novo método createCriteria(...) que devolve um jakarta.persistence.criteria.CriteriaQuery, usando o CriteriaBuilder do Session.

A partir do entityName (nome de entidade cadastrado no metamodel), recuperamos a classe da entidade (getMetamodel().entity(...).getJavaType()) e iniciamos a consulta com builder.createQuery() e criteria.from().

3. Imports e Assinaturas

Removido import org.hibernate.FlushMode; e substituído por import jakarta.persistence.FlushModeType;.

Inserido import de jakarta.persistence.criteria.*.

Ajuste na assinatura do método createCriteria: agora é genérico em e retorna CriteriaQuery em vez de Criteria.

4. Compatibilidade de Sessão

A nova createCriteria só suporta sessão stateful (Session), lançando exceção para StatelessSession. Se você precisar de critérios em sessão stateless, considere outro mecanismo (ex.: HQL).

promt

Estou migrando minha aplicacao Java do Hibernate 5 para o Hibernate 6.1.7.Final. Considerando a lista de erros de compilacao a seguir reescreva a classe Java mais abaixo, mantendo os nomes de variaveis, métodos e objetos originais, adaptando a para que funcione com o Hibernate 6.1.7.Final. Ao final apresente um relatório com as diferencas entre o codigo original e o adaptado (ao estilo Diff) e explique as alteracoes feitas.

Lista de Erros:

Classe Java a ser corrigida:

Analise a lista de erros de compilacao abaixo e corrija os erros na classe de testes de forma que ela compile e rode corretamente. A classe que contém os métodos testados também está anexa. Mantenha nomes de variaveis, métodos e objetos inalterados.

Lista de Erros:

Classe Java que contém os métodos testados:

Classe Java que contém os testes a serem orrigidos

org.hibernate hibernate-tools 6.1.7.Final