GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport
ceedcv CENTRE ESPECÍFIC D'EDUCACIÓ A DISTÀNCIA DE LA COMUNITAT VALENCIANA

Unidad 3. ACCESO UTILIZANDO MAPEO RELACIONAL DE OBJETOS (ORM)

Parte 3. Hibernar usando HQL

Acceso a Datos (ADA) (a distancia en inglés)
CFGS Desarrollo de Aplicaciones Multiplataforma (DAM)

Abelardo Martinez
Año 2023-2024

Créditos

•Apuntes realizados por Abelardo Martínez. •Basado y modificado de Sergio Badal (www.sergiobadal.com). •Las imágenes e iconos utilizados están protegidos por la LGPL . licencia y haber sido de:

- •https://commons.wikimedia.org/wiki/Crystal_Clear
- •https://www.openclipart.org

Progreso de la unidad

Contenido

1. CONSULTAS DE HIBERNACIÓN

Hibernar consultas

Hibernate proporciona varias opciones para tratar nuestra base de datos:

1) Consultas SQL nativas (bajo nivel de abstracción) • Utilice
PRINCIPALMENTE lenguaje SQL nativo para establecer consultas complejas

2) Consultas HQL (nivel medio de abstracción) •MEZCLAR
lenguaje y métodos de consulta de Hibernate específicos

3) Consultas HQL utilizando criterios (alto nivel de abstracción) • Utilice
PRINCIPALMENTE métodos específicos (¡SIN NINGUNA CONSULTA!)

2. USO DE CONSULTAS NATIVAS

Hibernar consultas nativas

• Hibernate proporciona una opción para ejecutar consultas SQL nativas mediante el uso del objeto SQLQuery. •
Hibernate SQL Query es muy útil cuando tenemos que ejecutar consultas específicas del proveedor de bases de datos que no son compatibles con la API de
Hibernate. • Para escenarios normales, la consulta SQL de hibernación no es el enfoque recomendado porque perdemos beneficios relacionados con la asociación de hibernación y la caché de primer nivel de hibernación.

• La sintaxis es la siguiente:

```
consulta = session.createSQLQuery("seleccione ... desde... donde... ordene por...");
filas = consulta.lista();
para (Objeto [] fila: filas) {...}
```

Ejemplo detallado: https://www.journaldev.com/3422/hibernate-native-sql-query-example

Ejemplo 1

Por ejemplo, puede unir dos tablas mediante consultas SQL directas:

```
consulta = session.createSQLQuery("seleccione e.emp_id, emp_name, emp_salary,address_line1, ciudad,
    código postal del empleado e, dirección a donde a.emp_id=e.emp_id");
filas = consulta.lista();
for(Objeto[] fila: filas)
    { Empleado emp = nuevo Empleado();
    emp.setId(Long.parseLong(row[0].toString()));
    emp.setName(fila[1].toString());
    emp.setSalary(Double.parseDouble(row[2].toString()));
    Dirección dirección = nueva
    dirección();
    dirección.setAddressLine1(fila[3].toString());
    dirección.setCity(fila[4].toString());

    dirección.setZipcode(fila[5].toString()); emp.setAddress(dirección); System.out.println(emp);
}
```

Ejemplo 2

Además, puede agregar parámetros:

```
consulta =
        sesión .createSQLQuery("seleccione emp_id, emp_name, emp_salary de Empleado donde emp_id = ?");
Lista<Objeto[]> empData = query.setLong(0, 1L).list();
for (Objeto[] fila: empData)
    { Empleado emp = nuevo
    Empleado();

    emp.setId(Long.parseLong(row[0].toString()));
    emp.setName(fila[1].toString()); emp.setSalary(Double.parseDouble(row[2].toString())); System.out.println(emp);
}
```

Para resumir

- Esa fue una breve introducción de Hibernate Native SQL Query; debe evitar su uso a menos que desee ejecutar consultas específicas de bases de datos.

- Para obtener una visión detallada de esas consultas, puede consultar aquí: https://docs.jboss.org/hibernate/core/3.3/reference/en/html/querysql.html

3. CONSULTAS HQL

3.1 Uso de consultas HQL

Hibernar lenguaje de consulta (HQL)

- Hibernate Query Language (HQL) es un lenguaje de consulta orientado a objetos, similar a SQL, pero en lugar de operar en tablas y columnas, HQL trabaja con objetos persistentes y sus propiedades.
- Hibernate traduce las consultas HQL en consultas SQL convencionales, que a su vez realizan acciones en la base de datos.
- Aunque puede utilizar SQL nativo, se recomienda utilizar HQL siempre que sea posible para evitar problemas de portabilidad de la base de datos y aprovechar las estrategias de almacenamiento en caché de Hibernate.
- La sintaxis es la siguiente:

```
Listar elementos = session.createQuery("desde ... donde... ordenar por...").list();
for (Iterador iterador = items.iterator(); iterator.hasNext();) {…}
```

Tenga en cuenta que las palabras clave como SELECT, FROM y WHERE, etc., no distinguen entre mayúsculas y minúsculas, pero propiedades como los nombres de tablas y columnas sí distinguen entre mayúsculas y minúsculas en HQL.

SELECCIONAR DE

Utilizado hasta ahora en nuestro código (de la semana pasada)

Nuevo

DÓNDE / PEDIR

Nuevo

Nuevo

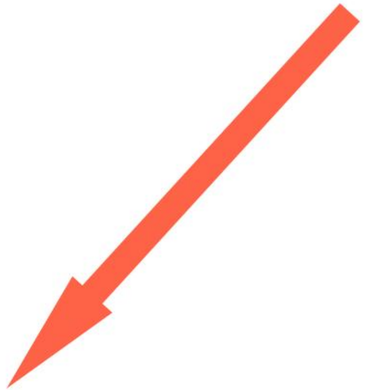
AGRUPAR POR

Nuevo

• También puede utilizar oraciones DML complejas (INSERT, UPDATE, DELETE) como puede consultar aquí: https://www.tutorialspoint.com/hibernate/hibernate_query_language.htm

FUNCIONES AGREGADAS

- Puede llamar a avg(), min(), max(), etc. funciones agregadas mediante HQL. • Veamos algunos ejemplos comunes:



Ser conscientes de la forma en que debemos obtener estos datos.

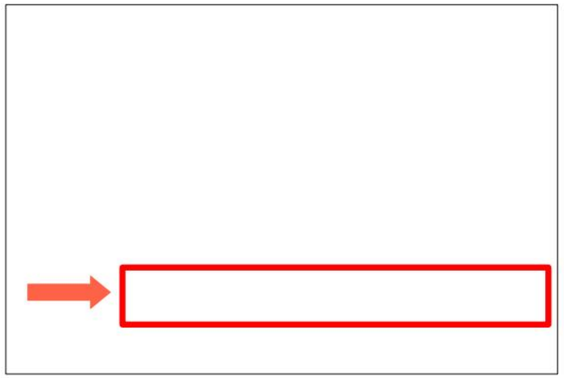
Para más información: https://www.javatpoint.com/hql

3.2 Ejemplo de actualización con HQL

Actualización a consultas HQL. programa principal

```
clase pública TestHibernateMySQL {

    // PROGRAMA
  PRINCIPAL public static void main(String[] stArgs) {

      //Crear nuevos objetos DAO para operaciones CRUD
      EmpleadoDAO objEmpleadoDAO = nuevo EmpleadoDAO();
      CertificateDAO objCertificateDAO = nuevo CertificateDAO();
      //TRUNCAR TABLAS. Eliminar todos los registros de las tablas
      objEmployeeDAO.deleteAllItems();
      objCertificateDAO.deleteAllItems();

      /* Agregar registros en la base de datos */
      Certificado objCert1 = objCertificateDAO.addCertificate("MBA"); Certificado objCert2 =
      objCertificateDAO.addCertificate("PMP");
      //Conjunto de certificados
      HashSet<Certificate> hsetCertificates = new HashSet<Certificate>();
      hsetCertificates.add(objCert1);
      hsetCertificates.add(objCert2);
      /* Agregar registros en la base de datos */
      Empleado objEmp1 = objEmployeeDAO.addEmployee("Alfred", "Vincent", 15000, hsetCertificates); Empleado objEmp2 =
      objEmployeeDAO.addEmployee("John", "Gordon", 8000, hsetCertificates); Empleado objEmp3 =
      objEmployeeDAO.addEmployee("Edgard", "Codd", 20000, hsetCertificates); Empleado objEmp4 =
      objEmployeeDAO.addEmployee("Joseph", "Smith", 5000, hsetCertificates); Empleado objEmp5 =
      objEmployeeDAO.addEmployee("Mary", "Jones", 15000, hsetCertificates);
      /* Enumera todos los empleados */
      objEmployeeDAO.listEmployees();
      objEmployeeDAO.listRichEmployees();

      //Cerrar la fábrica de sesiones de hibernación global
      HibernateUtil.shutdownSessionFactory();
  }
}
```

Ultimas semanas



De esta semana

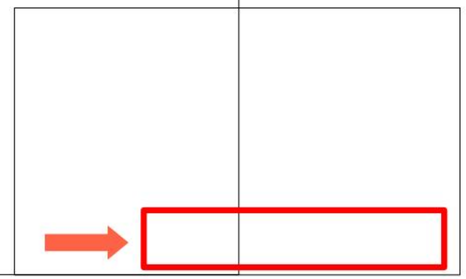
Crearemos un nuevo método para enumerar solo los empleados que ganan

> 10000, ordenados por descripción de salario.

Actualización a consultas HQL. Método

Es tan fácil como agregar la condición usando HQL, casi idéntico (en este caso) al SQL estándar.

```
/* Método para LEER todos los empleados que ganan > 10000 */ public
void listRichEmployees() {

    Sesión hibSession = HibernateUtil.SFACTORY.openSession(); //abre la fábrica de sesiones de hibernación Transacción txDB
    = null; //transacción de base de datos

    intente
        { txDB = hibSession.beginTransaction(); //inicia la transacción Lista<Empleado>
        listEmployees = hibSession.createQuery("DESDE Empleado DONDE dSalary > 10000 ORDEN POR dSalary", Empleado.class).list(); si (listaEmpleados.isEmpty())

            System.out.println("******** No se encontraron elementos");
        else
            System.out.println("\n***** Iniciar listado...\n");

        for (Iterador<Empleado> itEmployee = listEmployees.iterator(); itEmployee.hasNext();) {
            Empleado objEmpleado = (Empleado) itEmployee.next();
            System.out.print(" Nombre: " + objEmployee.getstFirstName() + " | "); System.out.print(" Apellido: "
            + objEmployee.getstLastName() + " | "); System.out.println("Salario: " + objEmployee.getdSalary());
            Set<Certificado> setCertificates = objEmployee.getrelCertificates(); for
            (Iterator<Certificado> itCertificate = setCertificates.iterator(); itCertificate.hasNext();) {

                Certificado objCertificate = (Certificado) itCertificate.next(); System.out.println("Certificado:
                " + objCertificate.getstCertName());
            }

        } txDB.commit(); //finaliza la transacción }
    catch (HibernateException hibe) {
        si (txDB ! = nulo)
            txDB.rollback(); //algo salió mal, así que revertir hibe.printStackTrace(); }
        finalmente { hibSession.close(); //
    cerrar sesión
        de hibernación
    }
}
```



4. CRITERIOS HQL

4.1 Uso de criterios HQL

Criterios HQL

- El lenguaje de consulta de criterios de Hibernación (HCQL) se utiliza para recuperar los registros según criterios específicos.
- La interfaz Criterios proporciona muchos métodos para especificar criterios.

El objeto de Hibernate Criteria createCriteria() está en desuso desde Hibernate 5 (septiembre de 2015), deberías usar JPA CriteriaQuery en su lugar.
https://stackoverflow.com/questions/40720799/deprecated-createcriteria-method-in-hibernate-5

- La mayoría de los recursos que encontrará navegando por la red serán sobre la versión obsoleta. Puede utilizar estos para utilizar el nuevo enfoque:

https://www.baeldung.com/hibernate-criteria-queries

Extensiones específicas de Hibernate para la API de Criteria

• La especificación JPA define el lenguaje de consulta JPQL basado en cadenas. •
Hibernate lo amplía (JPA) para admitir elementos como funciones específicas de bases de datos, funciones de ventanas y operaciones basadas en conjuntos.
• Desde la versión 6, Hibernate ha hecho lo mismo con la API Criteria de JPA. • La sintaxis es la siguiente:

```
CriteriaBuilder cb = sesión.getCriteriaBuilder();

CriteriaQuery<Item> cr = cb.createQuery(Item.class);
cr.select(root).where(cb.gt(root.<Integer>get("campo"), 10000));
cr.orderBy(cb.desc(root.get("campo")));

Consulta<Artículo> consulta = session.createQuery(cr);
Lista<Elemento> elementos =
query.getResultList(); for (Iterador iterador = items.iterator(); iterator.hasNext();) {…}
```

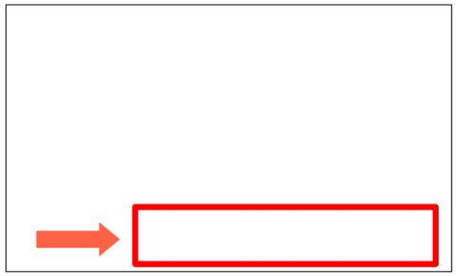
Para más información: https://thorben-janssen.com/hibernate-specific-extensions-to-the-criteria-api/

Actualización a los criterios HQL. Método

```
lista pública vacíaaEmpleadosUsandoCriteria () {

        Sesión hibSession = HibernateUtil.SFACTORY.openSession(); //abre la fábrica de sesiones de hibernación Transacción txDB
        = null; //transacción de base de datos System.out.println("\n*****
        Listado de empleados usando criterios HQL...\n");

        intente
              { txDB = hibSession.beginTransaction(); //inicia la transacción // 1. Cree una
              instancia de CriteriaBuilder llamando al método // Session.getCriteriaBuilder().

              CriteriaBuilder crbCritBuilder = hibSession.getCriteriaBuilder(); // 2. Cree un objeto de
              consulta creando una instancia de la interfaz CriteriaQuery //.

              CriteriaQuery<Empleado> crqHQL = crbCritBuilder.createQuery(Empleado.clase); // 3. Establezca la raíz
              de la consulta llamando al método from() en el objeto CriteriaQuery // para definir una variable de rango
              en la cláusula FROM.
              Raíz<Empleado> rootEmpleado = crqHQL.from(Empleado.clase); // 4.
              Especifique cuál será el tipo de resultado de la consulta llamando al método select() // del objeto CriteriaQuery.
              crqHQL.select(empleado raíz); // 5. Prepare la
              consulta para su ejecución creando
              una instancia org.hibernate.query.Query // llamando al método Session.createQuery(), especificando el
              tipo de resultado de la consulta.
              Consulta<Empleado> qryHQL = hibSession.createQuery(crqHQL); // 6.
              Ejecute la consulta llamando al método getResultList() o getSingleResult() // en el objeto
              org.hibernate.query.Query.
              Lista<Empleado> lstEmpleado = qryHQL.getResultList(); si
              (lstEmployee.isEmpty())
                    System.out.println("******** No se encontraron elementos");
              for (Iterator<Empleado> itEmployee = lstEmployee.iterator(); itEmployee.hasNext();) { Empleado objEmployee =
                    (Empleado) itEmployee.next(); System.out.print(" Nombre: " +
                    objEmployee.getstFirstName() + " | "); System.out.print(" Apellido: " + objEmployee.getstLastName()
                    + " | "); System.out.println("Salario: " + objEmployee.getdSalary()); Set<Certificado> relCertificates
                    = objEmployee.getrelCertificates(); for (Iterator<Certificado> itCertificate =
                    relCertificates.iterator(); itCertificate.hasNext();) {

                          Certificado objCertificate = (Certificado) itCertificate.next(); System.out.println("Certificado:
                          " + objCertificate.getstCertName());
                    }

              } txDB.commit(); //finaliza la transacción
        } catch (HibernateException hibe) { if (txDB ! =
              null) txDB.rollback(); //
                    algo salió mal, así que revertir hibe.printStackTrace(); } finalmente {


              hibSession.close(); //cerrar sesión de hibernación
        }
}
```

Estos son el estándar

instrucciones para obtener todos los elementos de una clase.



Operadores de comparación

Y estos, los refinamientos que podemos aplicar para obtener datos específicos.

… cb.gt(root.<Integer>get(“Precio del artículo”), 1000)...

Propiedad NULA/NO NULA

Y estos, los refinamientos que podemos aplicar para obtener datos específicos.

Operadores lógicos y expresiones en cadena.

Y estos, los refinamientos que podemos aplicar para obtener datos específicos.

Funciones agregadas

Y estos, los refinamientos que podemos aplicar para obtener datos específicos.

ORDENAR POR

Y estos, los refinamientos que podemos aplicar para obtener datos específicos.

También puede utilizar oraciones DML complejas (INSERT, UPDATE, DELETE) y utilizar predicados como puede consultar aquí: https://www.baeldung.com/hibernate-criteria-queries

4.2 Ejemplo de actualización con criterios HQL

Actualización a los criterios HQL. Método

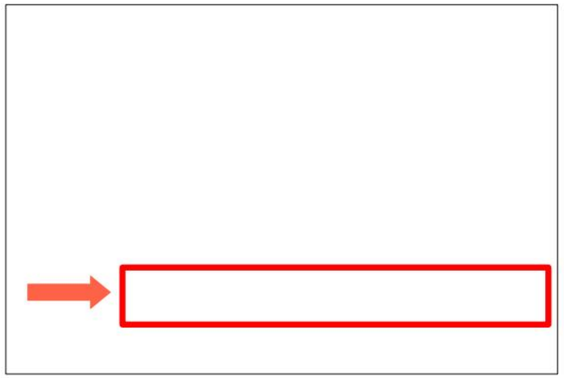
```
lista pública vacíaRichEmployeesUsingCriteria () {

        Sesión hibSession = HibernateUtil.SFACTORY.openSession(); //abre la fábrica de sesiones de hibernación Transacción txDB
        = null; //transacción de base de datos System.out.println("\n*****
        Listado de empleados usando criterios HQL...\n");

        intente
              { txDB = hibSession.beginTransaction(); //inicia la transacción // 1. Cree una
              instancia de CriteriaBuilder llamando al método // Session.getCriteriaBuilder().

              CriteriaBuilder crbCritBuilder = hibSession.getCriteriaBuilder(); // 2. Cree un objeto de
              consulta creando una instancia de la interfaz CriteriaQuery //.

              CriteriaQuery<Empleado> crqHQL = crbCritBuilder.createQuery(Empleado.clase); // 3. Establezca la raíz
              de la consulta llamando al método from() en el objeto CriteriaQuery // para definir una variable de rango
              en la cláusula FROM.
              Raíz<Empleado> rootEmpleado = crqHQL.from(Empleado.clase); // 4.
              Especifique cuál será el tipo de resultado de la consulta llamando al método select() // del objeto CriteriaQuery.

              crqHQL.select(rootEmployee).where(crbCritBuilder.gt(rootEmployee.<Integer>get("dSalary"), 10000));
              crqHQL.orderBy(crbCritBuilder.desc(rootEmployee.get("dSalary"))); // 5. Prepare la
              consulta para su ejecución creando una instancia org.hibernate.query.Query // llamando al método
              Session.createQuery(), especificando el tipo de resultado de la consulta.
              Consulta<Empleado> qryHQL = hibSession.createQuery(crqHQL); // 6.
              Ejecute la consulta llamando al método getResultList() o getSingleResult() // en el objeto
              org.hibernate.query.Query.
              Lista<Empleado> lstEmpleado = qryHQL.getResultList(); si
              (lstEmployee.isEmpty())
                    System.out.println("******** No se encontraron elementos");
              for (Iterator<Empleado> itEmployee = lstEmployee.iterator(); itEmployee.hasNext();) { Empleado objEmployee =
                    (Empleado) itEmployee.next(); System.out.print(" Nombre: " +
                    objEmployee.getstFirstName() + " | "); System.out.print(" Apellido: " + objEmployee.getstLastName()
                    + " | "); System.out.println("Salario: " + objEmployee.getdSalary()); Set<Certificado> relCertificates
                    = objEmployee.getrelCertificates(); for (Iterator<Certificado> itCertificate =
                    relCertificates.iterator(); itCertificate.hasNext();) {

                          Certificado objCertificate = (Certificado) itCertificate.next(); System.out.println("Certificado:
                          " + objCertificate.getstCertName());
                    }

              } txDB.commit(); //finaliza la transacción
        } catch (HibernateException hibe) { if (txDB ! =
              null) txDB.rollback(); //
                    algo salió mal, así que revertir hibe.printStackTrace(); } finalmente {


              hibSession.close(); //cerrar sesión de hibernación
        }
}
```

Crearemos un nuevo método.

para enumerar solo los empleados que ganan > 10000, ordenar por descripción de salario

Actualización a los criterios HQL. programa principal

```
clase pública TestHibernateMySQL {

    // PROGRAMA
    PRINCIPAL public static void main(String[] stArgs) {

        //Crear nuevos objetos DAO para operaciones CRUD
        EmpleadoDAO objEmpleadoDAO = nuevo EmpleadoDAO();
        CertificateDAO objCertificateDAO = nuevo CertificateDAO();

        //TRUNCAR TABLAS. Eliminar todos los registros de las tablas
        objEmployeeDAO.deleteAllItems();
        objCertificateDAO.deleteAllItems();

        /* Agregar registros en la base de datos */
        Certificado objCert1 = objCertificateDAO.addCertificate("MBA"); Certificado objCert2 =
        objCertificateDAO.addCertificate("PMP");

        //Conjunto de certificados
        HashSet<Certificate> hsetCertificates = new HashSet<Certificate>();
        hsetCertificates.add(objCert1);
        hsetCertificates.add(objCert2);

        /* Agregar registros en la base de datos */
        Empleado objEmp1 = objEmployeeDAO.addEmployee("Alfred", "Vincent", 15000, hsetCertificates); Empleado objEmp2 =
        objEmployeeDAO.addEmployee("John", "Gordon", 8000, hsetCertificates); Empleado objEmp3 =
        objEmployeeDAO.addEmployee("Edgard", "Codd", 20000, hsetCertificates); Empleado objEmp4 =
        objEmployeeDAO.addEmployee("Joseph", "Smith", 5000, hsetCertificates); Empleado objEmp5 =
        objEmployeeDAO.addEmployee("Mary", "Jones", 15000, hsetCertificates);

        /* Enumere todos los empleados */ //NUEVA
        MANERA (HQL usando criterios)
        objEmployeeDAO.listEmployeesUsingCriteria(); /* Enumere
        todos los empleados que ganan > 10000 */ //NUEVA MANERA
        (HQL usando criterios)
        objEmployeeDAO.listRichEmployeesUsingCriteria();

        //Cerrar la fábrica de sesiones de hibernación global
        HibernateUtil.shutdownSessionFactory();
    }
}
```



5. ACTIVIDADES PARA LA PRÓXIMA SEMANA

Actividades propuestas

Consulta las sugerencias de ejercicios que encontrarás en el “Aula Virtual”. Estas actividades son opcionales y no evaluables, pero comprenderlas es esencial para resolver la tarea evaluable que tenemos por delante.

En breve encontrará las soluciones propuestas.

6. BIBLIOGRAFÍA

Recursos

- Punto de tutoriales. Tutorial de hibernación. https://www.tutorialspoint.com/hibernate/index.htm
- Introducción a Hibernate 6.

https://docs.jboss.org/hibernate/orm/6.3/introduction/html_single/Hibernate_Introduction.html#queries

- Guía del usuario de Hibernate ORM 6.0.0.CR1.

https://docs.jboss.org/hibernate/orm/6.0/userguide/html_single/Hibernate_User_Guide.html#pc

- Guía de migración de Hibernate 6.0.

https://docs.jboss.org/hibernate/orm/6.0/migration-guide/migration-guide.html

- Josep Cañellas Bornas, Isidre Guixà Miranda. Accés a dades. Desarrollo de aplicaciones multiplataforma. Comunes creativos. Departamento de Enseñanza, Institut Obert de Catalunya. Dipósito legal: B. 29430-2013. https://ioc.xtec.cat/educacio/recursos

- Alberto Oliva Molina. Acceso a datos. UD 3. Herramientas de mapeo de objetos relacionales (ORM). IES Tubalcaín. Tarazona (Zaragoza, España).

GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport

ceedcv CENTRE ESPECÍFIC D'EDUCACIÓ A DISTÀNCIA DE LA COMUNITAT VALENCIANA