



theJavaGeek

y aprendizaje automático con python también :)

JAVA EE , JPA 2

Ejemplo de JPA OrderColumn

por [Prasad Kharkar](#) • 24 de septiembre de 2014 • [1 comentario](#)

Bio

últimas publicaciones



Prasad Kharkar

Prasad Kharkar es un entusiasta de Java y siempre está interesado en explorar y aprender tecnologías Java. Él es SCJP, OCPWCD, OCEJPAD y aspira a ser arquitecto de Java.



Considere que queremos mantener el orden persistente de la lista cuando se inserta en la base de datos y queremos recuperar la lista en el orden en que se insertó. Este ejemplo de JPA OrderColumn demuestra cómo se puede hacer.

Ejemplo de JPA OrderColumn:

Veamos directamente lo que dice javadoc oficial sobre [OrderColumn](#)

Especifica una columna que se utiliza para mantener el orden persistente de una lista. El proveedor de persistencia es responsable de mantener el orden después de la recuperación y en la base de datos. El proveedor de persistencia es responsable de actualizar el orden en la descarga de la base de datos para reflejar cualquier inserción, eliminación o reordenación que afecte a la lista.

Entidades para JPA OrderColumn Ejemplo:

Considere que tenemos una entidad de **empleado** con los campos **idEmployee** , **firstName** , **lastName** , **salary** y **phones** donde phones es una lista de cadenas. Podemos usar [ElementCollection](#) para almacenar cadenas. Entonces la entidad será de la siguiente manera.

```
package com.thejavageek.jpa.entities;

import java.util.List;

import javax.persistence.ElementCollection;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
```

```
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OrderColumn;
import javax.persistence.TableGenerator;

@Entity
public class Employee {

    @TableGenerator(name = "employee_gen", table = "ID_GEN", pkColumnName = "GEN_NAME", valueCol
    @Id
    @GeneratedValue(strategy = GenerationType.TABLE, generator = "employee_gen")
    private int idEmployee;
    private String firstName;
    private String lastName;
    private int salary;

    @OrderColumn(name = "phone_sequence")
    @ElementCollection
    private List<String> phones;

    public int getIdEmployee() {
        return idEmployee;
    }

    public void setIdEmployee(int idEmployee) {
        this.idEmployee = idEmployee;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

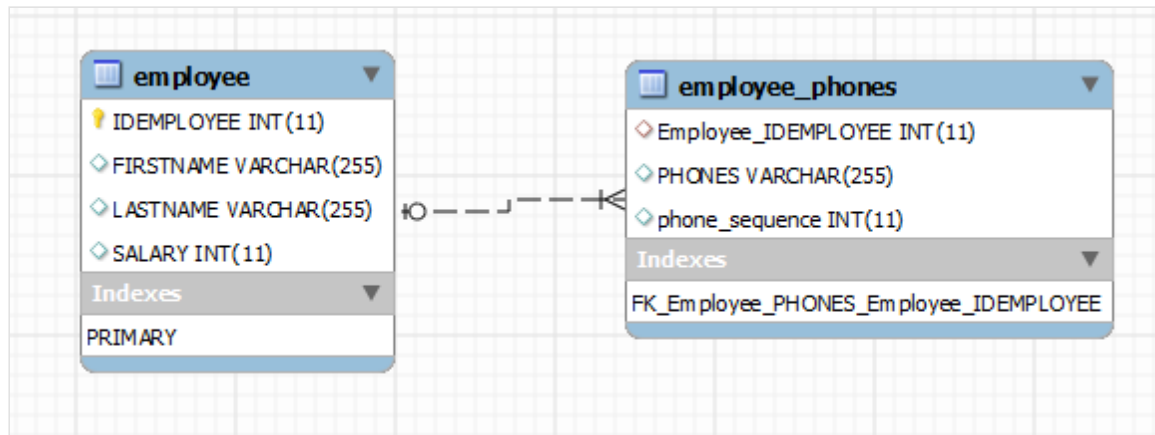
    public int getSalary() {
        return salary;
    }

    public void setSalary(int salary) {
        this.salary = salary;
    }

    public List<String> getPhones() {
        return phones;
    }

    public void setPhones(List<String> phones) {
        this.phones = phones;
    }
}
```

Como usamos **ElementCollection** en JPA OrderColumn Example, habrá una tabla separada para almacenar teléfonos. La estructura de la base de datos será la siguiente



Ahora que la base de datos y las entidades se crean para el ejemplo de JPA OrderColumn, probemoslo,

Java

```
public static EntityManagerFactory emf;
public static EntityManager em;

public static void main(String[] args) {

    emf = Persistence.createEntityManagerFactory("theJavaGeekJPA");
    em = emf.createEntityManager();

    EntityTransaction txn = em.getTransaction();
    txn.begin();

    Employee emp = new Employee();
    emp.setFirstName("Prasad");
    emp.setLastName("Kharkar");

    List<String> phones = new ArrayList<String>();
    phones.add("353745415");
    phones.add("225765367");
    phones.add("92573626726");
    emp.setPhones(phones);

    em.persist(emp);
    txn.commit();

    int id = emp.getIdEmployee();

    emp = em.find(Employee.class, id);

    phones = emp.getPhones();

    for (String s : phones) {
        System.out.println(s);
    }

    em.close();
    emf.close();
}
```

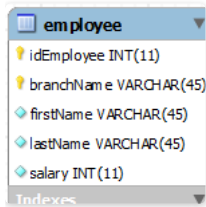
Después de ejecutar esto, puede ver que los números de teléfono se recuperan en el orden en que fueron insertados.

```
353745415
225765367
```

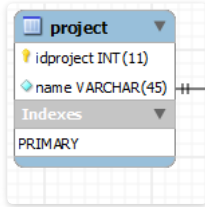
Espero que el ejemplo de JPA OrderColumn me haya ayudado. Feliz aprendizaje 😊



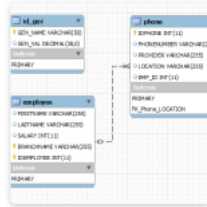
Artículos Relacionados



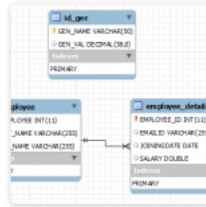
Ejemplo de JPA
[@EmbeddedId](#)



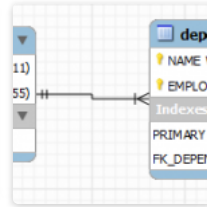
JPA mapeo
[muchos a muchos](#)



Ejemplo de
[anotación JPA
JoinColumns](#)



Ejemplo de
[anotación de tabla
secundaria JPA](#)



Ejemplo de JPA
[MapsId](#)

← Ejemplo de anotación JPA JoinColumns

Ejemplo de almacenamiento en caché JPA →

1 comentario para " Ejemplo de JPA OrderColumn "



Nagesh

27 de mayo de 2016 a las 11:17 a.m.

Según el ejemplo anterior, ¿podría decirme cómo podemos eliminar los datos de la tabla employee_phones? Si eliminamos un registro específico de employee_phones, ¿volverá a realizarse el pedido en la columna phone_sequence?

Por favor hagame saber.

Respuesta

Deja una respuesta

Su dirección de correo electrónico no será publicada. Los campos obligatorios están marcados *

Comentario

Nombre *

Correo electrónico *

Sitio web

☐ Guarde mi nombre, correo electrónico y sitio web en este navegador para la próxima vez que comento.

publicar comentario





Openbank
Grupo Santander

Cuenta Corriente


SIN
COMISIONES

CONSIGUE TU
CHEQUE REGALO DE
amazon.es
DE
30€

* Consulta condiciones de uso
válida hasta el 21/05/2018
para los 1.000 primeros
clientes. Amazon.es participa
en esta promoción.
Se aplican restricciones.

Hazte

Search ...



TheJavaGeek

Like Page

1.6K likes





Copyright © 2018 theJavaGeek . Todos los derechos reservados.
The Magazine Basic Theme por bavotasan.com .