ConsultasJPQL

```
//1. Consulta de todos los objetos de tipo Persona
log.debug(message:"\n1. Consulta de todas las Personas");
jpql = "select p from Persona p";
personas = em.createQuery(qlString:jpql).getResultList();
mostrarPersonas( personas );
```

```
1. Consulta de todas las Personas
[EL Fine]: sql: 2023-11-02 20:24:34.979--ServerSession(156710276)--Connection(115086468)--SELECT id_persona, APELLIDO, EMAIL, NOMBRE, TELEFONO FROM PERSONA
20:24:35 [main] DEBUG  - Persona{idPersona=1, nombre=Juan, apellido=Perez, email=jperez@mail.com, telefono=22577777}
20:24:35 [main] DEBUG  - Persona{idPersona=2, nombre=Karla, apellido=Gomez, email=kgomez@mail.com, telefono=25252525}
20:24:35 [main] DEBUG  - Persona{idPersona=3, nombre=Maria, apellido=Gutierrez, email=mgutierrez@mail.com, telefono=88991199}
20:24:35 [main] DEBUG  - Persona{idPersona=4, nombre=Pedro, apellido=Luna, email=pedro@gmail.com, telefono=13132153}
20:24:35 [main] DEBUG  - Persona{idPersona=6, nombre=Hugo, apellido=Hernandez, email=hhernandez@mail.com, telefono=55778822}
------------------------------------------------------------------------
```

```
//2. Consula de la Persona con id = 1
log.debug(message:"\n2. consulta de la Persona con id = 1");
jpql = "select p from Persona p where p.idPersona = 1";
persona = (Persona) em.createQuery(qlString:jpql).getSingleResult();
log.debug(message:persona);
```

```
2. consulta de la Persona con id = 1
[EL Fine]: sql: 2023-11-02 20:27:07.892--ServerSession(53940034)--Connection(56510351)--SELECT id_persona, APELLIDO, EMAIL, NOMBRE, TELEFONO FROM PERSONA WHERE (id_persona = ?)
        bind => [1 parameter bound]
20:27:07 [main] DEBUG  - Persona{idPersona=1, nombre=Juan, apellido=Perez, email=jperez@mail.com, telefono=22577777}
------------------------------------------------------------------------
```

```
//3. Consulta de la Persona por nombre
jpql = "select p from Persona p where p.nombre = 'Karla'";
personas = em.createQuery(qlString:jpql).getResultList();
mostrarPersonas(personas);
```

```
[EL Fine]: sql: 2023-11-02 20:27:52.398--ServerSession(53940034)--Connection(144467413)--SELECT id_persona, APELLIDO, EMAIL, NOMBRE, TELEFONO FROM PERSONA WHERE (NOMBRE = ?)
        bind => [1 parameter bound]
20:27:52 [main] DEBUG  - Persona{idPersona=2, nombre=Karla, apellido=Gomez, email=kgomez@mail.com, telefono=25252525}
------------------------------------------------------------------------
```

```
//4. Consulta de datos individuales, se crea un arreglo(tupla) de tipo object de 3 columnas
log.debug(message:"\n4. Consulta de datos individuales, se crea un arreglo (tupla) de tipo object de 3 columnas");
jpql = "select p.nombre as Nombre, p.apellido as Apellido, p.email as Email from Persona p";
iter = em.createQuery(qlString:jpql).getResultList().iterator();
while(iter.hasNext()){
    tupla = (Object[]) iter.next();
    String nombre = (String) tupla[0];
    String apellido = (String) tupla[1];
    String email = (String) tupla[2];
    log.debug("nombre:" + nombre + ", apellido:" + apellido + ", email:" + email) ;
}
```

```
4. Consulta de datos individuales, se crea un arreglo (tupla) de tipo object de 3 columnas
[EL Fine]: sql: 2023-11-02 20:29:27.58--ServerSession(53940034)--Connection(1075996552)--SELECT NOMBRE, APELLIDO, EMAIL FROM PERSONA
20:29:27 [main] DEBUG  - nombre:Juan, apellido:Perez, email:jperez@mail.com
20:29:27 [main] DEBUG  - nombre:Karla, apellido:Gomez, email:kgomez@mail.com
20:29:27 [main] DEBUG  - nombre:Maria, apellido:Gutierrez, email:mgutierrez@mail.com
20:29:27 [main] DEBUG  - nombre:Pedro, apellido:Luna, email:pedro@gmail.com
20:29:27 [main] DEBUG  - nombre:Hugo, apellido:Hernandez, email:hhernandez@mail.com
------------------------------------------------------------------------
```

```java
//5. Obtiene el objeto Persona y el id, se crea un arreglo de tipo Object con 2 columnas
log.debug(message:"\n. Obtiene el objeto Persona y el id, se crea un arreglo de tipo Object con 2 columnas");
jpql = "select p, p.idPersona from Persona p ";
iter = em.createQuery(qlString: jpql).getResultList().iterator();
while(iter.hasNext()){
    tupla = (Object[]) iter.next();
    persona = (Persona) tupla[0];
    int idPersona = (int) tupla[1];
    log.debug("Objeto persona:" + persona);
    log.debug("id persona:" + idPersona );
}
```

```
20:30:16 [main] DEBUG  - Objeto persona:Persona{idPersona=1, nombre=Juan, apellido=Perez, email=jperez@mail.com, telefono=22577777}
20:30:16 [main] DEBUG  - id persona:1
20:30:16 [main] DEBUG  - Objeto persona:Persona{idPersona=2, nombre=Karla, apellido=Gomez, email=kgomez@mail.com, telefono=25252525}
20:30:16 [main] DEBUG  - id persona:2
20:30:16 [main] DEBUG  - Objeto persona:Persona{idPersona=3, nombre=Maria, apellido=Gutierrez, email=mgutierrez@mail.com, telefono=88991199}
20:30:16 [main] DEBUG  - id persona:3
20:30:16 [main] DEBUG  - Objeto persona:Persona{idPersona=4, nombre=Pedro, apellido=Luna, email=pedro@gmail.com, telefono=13132153}
20:30:16 [main] DEBUG  - id persona:4
20:30:16 [main] DEBUG  - Objeto persona:Persona{idPersona=6, nombre=Hugo, apellido=Hernandez, email=hhernandez@mail.com, telefono=55778822}
20:30:16 [main] DEBUG  - id persona:6
------------------------------------------------------------------------
```

```java
//6. Consulta de todas las personas
System.out.println(x: "\n6. Consulta de todas las personas");
jpql = "select new sv.com.gm.sga.domain.Persona( p.idPersona ) from Persona p";
personas = em.createQuery(qlString: jpql).getResultList();
mostrarPersonas(personas);
```

```
6. Consulta de todas las personas
[EL Fine]: sql: 2023-11-02 20:31:48.76--ServerSession(53940034)--Connection(1012552887)--SELECT id_persona FROM PERSONA
20:31:48 [main] DEBUG  - Persona{idPersona=1, nombre=null, apellido=null, email=null, telefono=null}
20:31:48 [main] DEBUG  - Persona{idPersona=2, nombre=null, apellido=null, email=null, telefono=null}
20:31:48 [main] DEBUG  - Persona{idPersona=3, nombre=null, apellido=null, email=null, telefono=null}
20:31:48 [main] DEBUG  - Persona{idPersona=4, nombre=null, apellido=null, email=null, telefono=null}
20:31:48 [main] DEBUG  - Persona{idPersona=6, nombre=null, apellido=null, email=null, telefono=null}
------------------------------------------------------------------------
```

```java
//7. Regresa el valor minimo y maxico del idPersona (scaler result)
System.out.println(x: "\n7. Regresa el valor minimo y maxico del idPersona (scaler result)");
jpql = "select min(p.idPersona) as MinId, max(p.idPersona) as MaxId, count(p.idPersona) as Contador from Persona p";
iter = em.createQuery(qlString: jpql).getResultList().iterator();
while(iter.hasNext()){
    tupla = (Object[]) iter.next();
    Integer idMin = (Integer) tupla[0];
    Integer idMax = (Integer) tupla[1];
    Long count = (Long) tupla[2];
    log.debug("idMin:" + idMin + ", idMax:" + idMax + ", count:" + count);
}
```

```
7. Regresa el valor minimo y maxico del idPersona (scaler result)
[EL Fine]: sql: 2023-11-02 20:32:27.81--ServerSession(53940034)--Connection(1243102466)--SELECT MIN(id_persona), MAX(id_persona), COUNT(id_persona) FROM PERSONA
20:32:27 [main] DEBUG  - idMin:1, idMax:6, count:5
------------------------------------------------------------------------
```

```java
//8. Cuenta los nombres de las personas que son distintos
log.debug(message:"\n8. Cuenta los nombres de las personas que son distintos");
jpql = "select count(distinct p.nombre) from Persona p";
Long contador = (Long) em.createQuery(qlString: jpql).getSingleResult();
log.debug("no. de personas con nombre distinto:" + contador);
```

```
8. Cuenta los nombres de las personas que son distintos
[EL Fine]: sql: 2023-11-02 20:34:34.401--ServerSession(53940034)--Connection(749693202)--SELECT COUNT(DISTINCT(NOMBRE)) FROM PERSONA
20:34:34 [main] DEBUG  - no. de personas con nombre distinto:5
------------------------------------------------------------------------
```

```java
log.debug(message:"\n9. Concatena y convierte a mayusculas el nombre y apellido");
jpql = "select CONCAT(p.nombre, ' ', p.apellido) as Nombre from Persona p";
nombres = em.createQuery(qlString: jpql).getResultList();
for(String nombreCompleto: nombres){
    log.debug(message:nombreCompleto);
}
```

```
9. Concatena y convierte a mayusculas el nombre y apellido
[EL Fine]: sql: 2023-11-02 20:34:57.026--ServerSession(53940034)--Connection(2015007762)--SELECT CONCAT(CONCAT(NOMBRE, ?), APELLIDO) FROM PERSONA
        bind => [1 parameter bound]
20:34:57 [main] DEBUG  - Juan Perez
20:34:57 [main] DEBUG  - Karla Gomez
20:34:57 [main] DEBUG  - Maria Gutierrez
20:34:57 [main] DEBUG  - Pedro Luna
20:34:57 [main] DEBUG  - Hugo Hernandez
------------------------------------------------------------------------
```

```java
//10. Obtiene el objeto persona con id igual al parametro proporcionado
log.debug(message:"\n10. Obtiene el objeto persona con id igual al parametro proporcionado");
int idPersona = 4;
jpql = "select p from Persona p where p.idPersona = :id";
q = em.createQuery(ql5tring: jpql);
q.setParameter(name: "id", value: idPersona);
persona = (Persona) q.getSingleResult();
log.debug(message:persona);
```

```
20:36:51 [main] DEBUG  -
10. Obtiene el objeto persona con id igual al parametro proporcionado
[EL Fine]: sql: 2023-11-02 20:36:51.687--ServerSession(53940034)--Connection(1031586763)--SELECT id_persona, APELLIDO, EMAIL, NOMBRE, TELEFONO FROM PERSONA WHER
        bind => [1 parameter bound]
20:36:51 [main] DEBUG  - Persona{idPersona=4, nombre=Pedro, apellido=Luna, email=pedro@gmail.com, telefono=13132153}
------------------------------------------------------------------------
```

```java
//11. Obtiene las personas que contengan una letra a en el nombre, sin importar si es mayusculas o minuscula
log.debug(message:"\n11. Obtiene las personas que contengan una letra a en el nombre, sin importar si es mayusculas o minuscula");
jpql = "select p from Persona p where upper(p.nombre) like upper(:parametro)";
String parametroString = "%a%";//es el caracter que utilizamos para el like
q = em.createQuery(ql5tring: jpql);
q.setParameter(name:"parametro", value: parametroString);
personas = q.getResultList();
mostrarPersonas(personas);
```

```
20:39:03 [main] DEBUG  -
11. Obtiene las personas que contengan una letra a en el nombre, sin importar si es mayusculas o minuscula
[EL Fine]: sql: 2023-11-02 20:39:03.512--ServerSession(53940034)--Connection(399653041)--SELECT id_persona, APELLIDO, EMAIL, NOMBRE, TELEFONO FROM PERSONA WHERE
        bind => [1 parameter bound]
20:39:03 [main] DEBUG  - Persona{idPersona=1, nombre=Juan, apellido=Perez, email=jperez@mail.com, telefono=22577777}
20:39:03 [main] DEBUG  - Persona{idPersona=2, nombre=Karla, apellido=Gomez, email=kgomez@mail.com, telefono=25252525}
20:39:03 [main] DEBUG  - Persona{idPersona=3, nombre=Maria, apellido=Gutierrez, email=mgutierrez@mail.com, telefono=88991199}
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  3.077 s
Finished at: 2023-11-02T20:39:03-06:00
------------------------------------------------------------------------
```

```java
//12. Uso de between
log.debug(message:"\n12. Uso de between");
jpql = "select p from Persona p where p.idPersona between 1 and 10";
personas = em.createQuery(ql5tring: jpql).getResultList();
mostrarPersonas(personas);
```

```
12. Uso de between
[EL Fine]: sql: 2023-11-02 20:39:55.849--ServerSession(53940034)--Connection(1803361784)--SELECT id_persona, APELLIDO, EMAIL, NOMBRE, TELEFONO FROM PERSONA WH
        bind => [2 parameters bound]
20:39:55 [main] DEBUG  - Persona{idPersona=1, nombre=Juan, apellido=Perez, email=jperez@mail.com, telefono=22577777}
20:39:55 [main] DEBUG  - Persona{idPersona=2, nombre=Karla, apellido=Gomez, email=kgomez@mail.com, telefono=25252525}
20:39:55 [main] DEBUG  - Persona{idPersona=3, nombre=Maria, apellido=Gutierrez, email=mgutierrez@mail.com, telefono=88991199}
20:39:55 [main] DEBUG  - Persona{idPersona=4, nombre=Pedro, apellido=Luna, email=pedro@gmail.com, telefono=13132153}
20:39:55 [main] DEBUG  - Persona{idPersona=6, nombre=Hugo, apellido=Hernandez, email=hhernandez@mail.com, telefono=55778822}
------------------------------------------------------------------------
```

```java
//13. Uso del ordenamiento
log.debug(message:"\n13. Uso del ordenamiento");
jpql = "select p from Persona p where p.idPersona > 1 order by p.nombre desc, p.apellido desc";
personas = em.createQuery(qlString:jpql).getResultList();
mostrarPersonas(personas);
```

```
13. Uso del ordenamiento
[EL Fine]: sql: 2023-11-02 20:40:43.922--ServerSession(53940034)--Connection(1047477166)--SELECT id_persona, APELLIDO, EMAIL, NOMBRE, TELEFONO FROM PERSONA WHERE
       bind => [1 parameter bound]
20:40:43 [main] DEBUG  - Persona{idPersona=4, nombre=Pedro, apellido=Luna, email=pedro@gmail.com, telefono=13132153}
20:40:43 [main] DEBUG  - Persona{idPersona=3, nombre=Maria, apellido=Gutierrez, email=mgutierrez@mail.com, telefono=88991199}
20:40:43 [main] DEBUG  - Persona{idPersona=2, nombre=Karla, apellido=Gomez, email=kgomez@mail.com, telefono=25252525}
20:40:43 [main] DEBUG  - Persona{idPersona=6, nombre=Hugo, apellido=Hernandez, email=hhernandez@mail.com, telefono=55778822}
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  2.964 s
Finished at: 2023-11-02T20:40:43-06:00
------------------------------------------------------------------------
```

```java
//14. Uso de subquery
log.debug(message:"\n14. Uso de subquery");
jpql = "select p from Persona p where p.idPersona in (select min(p1.idPersona) from Persona p1)";
personas = em.createQuery(qlString:jpql).getResultList();
mostrarPersonas(personas);
```

```
14. Uso de subquery
[EL Fine]: sql: 2023-11-02 20:41:19.66--ServerSession(53940034)--Connection(1002911155)--SELECT t0.id_persona, t0.APELLIDO, t0.EMAIL, t0.NOMBRE, t0.TELEFONO FROM PE
20:41:19 [main] DEBUG  - Persona{idPersona=1, nombre=Juan, apellido=Perez, email=jperez@mail.com, telefono=22577777}
```

```java
//15. Uso de join con lazy loading
log.debug(message:"\n15. Uso de join con lazy loading");
jpql = "select u from Usuario u join u.persona p";
usuarios = em.createQuery(qlString:jpql).getResultList();
mostrarUsuarios(usuarios);
```

```
15. Uso de join con lazy loading
[EL Fine]: sql: 2023-11-02 20:42:13.072--ServerSession(53940034)--Connection(557705922)--SELECT t1.id_usuario, t1.PASSWORD, t1.USERNAME, t1.id_persona FROM PERSONA t0, USUARIO t1 WHERE (t0.id_persona = t1.id_p
[EL Fine]: sql: 2023-11-02 20:42:13.096--ServerSession(53940034)--Connection(557705922)--SELECT id_persona, APELLIDO, EMAIL, NOMBRE, TELEFONO FROM PERSONA WHERE (id_persona = ?)
       bind => [1 parameter bound]
[EL Fine]: sql: 2023-11-02 20:42:13.099--ServerSession(53940034)--Connection(557705922)--SELECT id_persona, APELLIDO, EMAIL, NOMBRE, TELEFONO FROM PERSONA WHERE (id_persona = ?)
       bind => [1 parameter bound]
[EL Fine]: sql: 2023-11-02 20:42:13.1--ServerSession(53940034)--Connection(557705922)--SELECT id_persona, APELLIDO, EMAIL, NOMBRE, TELEFONO FROM PERSONA WHERE (id_persona = ?)
       bind => [1 parameter bound]
[EL Fine]: sql: 2023-11-02 20:42:13.101--ServerSession(53940034)--Connection(557705922)--SELECT id_persona, APELLIDO, EMAIL, NOMBRE, TELEFONO FROM PERSONA WHERE (id_persona = ?)
       bind => [1 parameter bound]
20:42:13 [main] DEBUG  - Usuario{idUsuario=1, username=jperez, password=123, persona=Persona{idPersona=1, nombre=Juan, apellido=Perez, email=jperez@mail.com, telefono=22577777}}
20:42:13 [main] DEBUG  - Usuario{idUsuario=2, username=kgomez, password=123, persona=Persona{idPersona=2, nombre=Karla, apellido=Gomez, email=kgomez@mail.com, telefono=25252525}}
20:42:13 [main] DEBUG  - Usuario{idUsuario=3, username=mgutierrez, password=123, persona=Persona{idPersona=3, nombre=Maria, apellido=Gutierrez, email=mgutierrez@mail.com, telefono=88991199}}
20:42:13 [main] DEBUG  - Usuario{idUsuario=4, username=hhernandez, password=123, persona=Persona{idPersona=6, nombre=Hugo, apellido=Hernandez, email=hhernandez@mail.com, telefono=55778822}}
------------------------------------------------------------------------
```

```java
    //16. Uso de left join con eager loading
    log.debug(message:"16. Uso de left join con eager loading");
    jpql = "select u from Usuario u left join fetch u.persona";
    usuarios = em.createQuery(qlString:jpql).getResultList();
    mostrarUsuarios(usuarios);
}
```

```
20:42:54 [main] DEBUG  - 16. Uso de left join con eager loading
[EL Fine]: sql: 2023-11-02 20:42:54.628--ServerSession(53940034)--Connection(1096411163)--SELECT t1.id_usuario, t1.PASSWORD, t1.USERNAME, t1.id_persona, t0.id_persona, t0.APELLIDO, t0.EMAIL, t0.NOMBRE, t0.TELEFONO F
20:42:54 [main] DEBUG  - Usuario{idUsuario=1, username=jperez, password=123, persona=Persona{idPersona=1, nombre=Juan, apellido=Perez, email=jperez@mail.com, telefono=22577777}}
20:42:54 [main] DEBUG  - Usuario{idUsuario=2, username=kgomez, password=123, persona=Persona{idPersona=2, nombre=Karla, apellido=Gomez, email=kgomez@mail.com, telefono=25252525}}
20:42:54 [main] DEBUG  - Usuario{idUsuario=3, username=mgutierrez, password=123, persona=Persona{idPersona=3, nombre=Maria, apellido=Gutierrez, email=mgutierrez@mail.com, telefono=88991199}}
20:42:54 [main] DEBUG  - Usuario{idUsuario=4, username=hhernandez, password=123, persona=Persona{idPersona=6, nombre=Hugo, apellido=Hernandez, email=hhernandez@mail.com, telefono=55778822}}
------------------------------------------------------------------------
BUILD SUCCESS
```