UNIVERSIDAD DE OVIEDO
EII – Grado IS – Repositorios de Información

Curso 2015 / 2016

Algunas pregunta interesantes que deberías saber responder

Implementar modelo en Java

Define Entidad y ValueType. ¿Qué es mutable e inmutable en cada uno de ellos? ¿Sobre qué atributos se debe deben definir los métodos *hashCode()* y *equals()* en cada uno de ellos?

¿Por qué escribimos métodos para el mantenimiento de las asociaciones *addXxxx(…)* y *removeXxxx(…)* en vez de hacer las asignaciones directamente?

¿Cómo se implementa en Java una clase asociativa UML?

¿Qué parámetros mínimos debe recibir el constructor de una Entidad?

¿Qué parámetros debe recibir el constructor de un ValueType?

¿Qué métodos no hay que implementar en un ValueType?

¿Por qué los ValueTypes deben ser inmutables?

¿Qué atributos no deben cambiar nunca en una Entidad?

En la implementación de una entidad ¿A qué llamábamos atributos naturales?

¿Para qué es útil redefinir el método *toString()* de las clases del modelo?

Funcionamiento del mapeador

Describe las diferencias entre los estados *Transient*, *Persistent* y *Detached*.

Un mapeador de objetos a relacional (O/R) debe resolver las diferencias entre los dos paradigmas.

- ¿Cómo resuelve un mapeador JPA la herencia?
- ¿Cómo resuelve la diferencia de granularidad?
- ¿Cómo resuelve la detección de la identidad?
- ¿Cómo resuelve la cardinalidad *MANY* to *MANY* de las asociaciones?
- ¿Cómo resuelve la navegación?
- ¿Cómo resuelve la concurrencia?

En un programa concurrente ¿cuantas copias parciales del grafo habrá en memoria?

¿Qué estrategias usan los mapeadores O/R para recrear en memoria una zona del grafo?

Para acelerar el acceso al grafo los mapeadores implementan una caché. ¿Qué características tiene esa caché?

En este orden: se crea una instancia del mapeador, se abre transacción, se recupera un objeto con el método *find()* y se invoca a unos cuántos métodos del objeto recuperado que le producen modificaciones. ¿Qué hay que hacer después para actualizar la base de datos con los cambios?

Describe brevemente las clases (o interfaces) principales que ofrece la API JPA: cuáles son, qué misión tienen, y cómo se interrelacionan.

¿Se puede navegar por el grafo una vez cerrado el contexto de persistencia? Matiza la respuesta.

¿A que puede ser debido que me salte una *LazyInitializationException*?

Mapeo

¿De qué formas se puede añadir información de mapeo en JPA?

Añadiendo anotaciones de mapeo, ¿cómo se vinculan los dos extremos de una asociación bidireccional?, ¿qué pasa si no se hace?

En JPA, con configuración de mapeo por defecto ¿los extremos *MANY* de una asociación se cargan de forma agresiva o bajo demanda?, ¿y los extremos *ONE*?

Un alumno se matrícula en varias asignaturas. En una asignatura se matricularán muchos alumnos. Para cada asignatura matriculada se obtiene una nota final. Dibuja el modelo UML. Escribe el código java mínimo para representar esa situación y las anotaciones de mapeo necesarias.

JPA, Hibernate, Eclipse Link. Explica qué relación hay entre esos términos.

Tenemos una aplicación desarrollada con mapeador JPA Hibernate funcionando contra una base de datos MySQL. Por alguna razón hay que cambiar de servidor de base de datos, ahora usaremos una Oracle. ¿Cómo lo resolvemos?

Consultas JPQL

¿De qué formas se pueden especificar y asignar los parámetros de una consulta?

¿Cómo se llama ese estilo de configuración de una *Query* a base de llamar a métodos que siempre devuelven el mismo objeto?

¿En qué estado están los objetos devueltos por una consulta JQL?, ¿y el que devuelve el método *find()*?

¿Qué fallos encuentras en estas consultas en JQL sobre *CarWorkshop*?

```
select * from TMecanicos m
```

```
select m from Mecanicos m
```

```
select m from mecanicos m
```

```
select * from TMecanicos
```

Sobre CarWorkshop ¿es correcta esta consulta? En caso negativo explica por qué no.

```
select a.vehiculo.cliente
from Averia a
where a.factura.fecha = '12/12/2014'
```

Sobre CarWorkshop ¿es correcta esta consulta? En caso negativo explica por qué no.

```
select f.averias.vehiculo.cliente
from Factura f
where f.fecha = '12/12/2014'
```

Arquitectura

Enumera los patrones que aparecen en la realización del CarWorkshop con mapeador.

Estoy escribiendo un comando en el que me están saliendo muchas líneas de código, ¿qué es muy probable que esté haciendo mal?

Imagina que el CarWorkshop que se ha desarrollado pide *login* y *password* a cada usuario que lo usa. Una vez verificada la identidad del usuario su *login* queda accesible en una variable global de hilo (un ThreadLocal). Ahora nos piden que modifiquemos el programa de forma que guarde en un fichero de log una línea <usuario, fecha, hora, método de la capa de servicio invocado> cada vez que un usuario realiza una acción (que guarde una auditoria). ¿En qué clase añadirías esa funcionalidad de forma que sólo se escriba una vez?

Suponiendo configuración de mapeo por defecto ¿funcionará el código mostrado a continuación? Explica los motivos.

Capa de presentación

```
AdminService as = ServicesFactory.adminServices();
Cliente c = as.findClientById( 123 );
for(Vehiculo v : c.getVehiculos()) {
    Printer.print( v );
}
...
```

Capa de servicio

```
class AdminServicesImpl {
   ...
   Cliente findClienteById(Long id) {
      EntityManager em = getEntityManagerAndOpenTrx();
      try {
          return em.find(Cliente.class, id);
      } finally {
         closeTrxAndEntityManager();
      }
   }
   ...
}
```

En el código de un comando: Tengo un objeto "Vehiculo" obtenido con un *find()* y necesito todas sus averías ¿cómo me hago con ellas? Comenta las posibilidades y señala la más sencilla.

En el código de una clase *Action*: Tengo un objeto "Vehiculo" obtenido con una llamada a la capa de lógica y necesito todas sus averías ¿cómo me hago con ellas? Comenta las posibilidades y señala la más sencilla.