Pregunta **13**

Respuesta guardada

Puntúa como 1,00

Si hay dos usuarios en concurrencia y ambos cargan zonas comunes del grafo, ¿cuántas copias de los objetos comunes hay a la vez en memoria con un mapeador que sigua la especificación JPA?

Seleccione una:

- a. Hay una copia por cada sesión de persistencia abierta, es decir, por cada usuario en concurrencia.
- b. Los objetos comunes sólo se instancian una vez, de ahí que haya que definir el hashCode() e equals() de las entidades sobre la identidad.
- c. Depende de la política de navegación que se escoja: eager o lazy. Con política agresiva los objetos comunes solo se cargan una vez, con la perezosa habrá copias individuales.
- d. La especificación lo deja a decisión de los implementadores del mapeador.

Quitar mi elección

Pregunta **14**

Respuesta guardada

Puntúa como 1,00

Si hay dos usuarios en concurrencia, ambos cargan zonas comunes del grafo y ambos modifican una entidad común, ¿cómo gestiona el mapeador la colisión de modificaciones?

Seleccione una:

- a. Dentro de cada sesión de persistencia se debe abrir una transacción y por tanto las modificaciones que se hagan están protegidas por los mecanismos de transacciones que ya conocemos.
- b. El usuario que haga la última actualización machacará lo que haya hecho el otro. El mapeador no tiene ningún control y eso es uno de los inconvenientes de trabajar con mapeadores. Debe ser el programador el que añada el código de control necesario.
- c. Es una situación tan poco habitual que la especificación ni siquiera se molesta en mencionarla.
- d. La situación se evita configurando la base de datos para que siempre tenga el nivel de aislamiento de las transacciones al máximo posible: SERIALIZABLE.

Quitar mi elección

Pregunta **15**

Respuesta guardada

Puntúa como 1,00

Imagina esta situación sobre la implementación de CarWorkshop que tenemos: Alice ejecuta el caso de uso "gestión de clientes" para actualizar los datos de uno porque el apellido está mal escrito. Bob, hace lo mismo unos segundos más tarde. En este momento tanto Alice como Bob están viendo los datos del (mismo) cliente en pantalla. Ahora, Alice modifica el apellido y justo cuando va a guardar los cambios le suena el teléfono y atiende la llamada. Mientras, a Bob le da tiempo a modificar el apellido y a añadir la fecha de nacimiento, que estaba sin rellenar, y guarda los cambios. Cuando Alice termina la llamada guarda sus cambios, varios minutos después de que Bob guardase los suyos. ¿Qué datos quedan en el sistema?

Seleccione una:

- a. Los de Alice
- b. Los de Bob
- c. Los de los dos combinados
- d. Ninguno de ellos

Quitar mi elección

Pregunta **16**

Respuesta guardada

Puntúa como 1,00

Sobre la situación anterior si analizamos paso a paso tenemos lo siguiente: Alice hace que se muestren en pantalla los datos del cliente, para ello provoca una transacción de lectura que dura lo que tarde el sistema en recuperar esos datos (milisegundos). Unos segundos más tarde Bob hace lo mismo. En este momento ambos están viendo lo mismo en pantalla. Cuando Bob guarda, se ejecuta una transacción de escritura que dura lo que tarde su proceso (milisegundos). Varios minutos más tarde Alice guarda sus datos en otra transacción de escritura, que no colisiona con ninguna otra. ¿Qué afirmación es cierta?

Seleccione una:

- a. Se dan a la vez dos tipos de transacciones: las de sistema, las que percibe y gestiona la base de datos y las de usuario, las que percibe el usuario (a veces se las llama transacciones de aplicación, o de negocio). Para el usuario editar el cliente es una sola operación, con independencia de su duración.
- b. Suele ser inviable hacer coincidir una transacción de usuario con una de sistema. Esto es porque la duración incontrolada de la transacción de usuario (imagina que se va a tomar un café, o no vuelve hasta el día siguiente...) dejaría bloqueado el sistema en lo que se refiere a ese cliente.
- c. Las transacciones de sistema (p.e. de base de datos) solo son capaces de evitar conflictos cuando las transacciones coinciden al mismo tiempo. Aquí no se da el caso. Debe ser la aplicación la que tome cartas en el asunto y se ocupe de gestionar expresamente la colisión en la transacción de usuario.
- d. Todas son ciertas.

Quitar mi elección

Pregunta **17**

Respuesta guardada

Puntúa como 1,00

Siguiendo aún con la situación anterior, ¿qué solución te parece más adecuada?

Seleccione una:

a. No hacer nada. Tampoco se va a dar esa situación de que vayan a coincidir a la vez, solo hay un Jefe de Taller. Y si se llegara a dar el caso de que hubiera dos, tampoco pasa nada porque uno de los dos repita la operación (si se dan cuenta, claro...).

b. Hacer que la aplicación mantenga una lista de bloqueos. Si un objeto está en esa lista significa que está bloqueado y que no se puede procesar. Así la operación de lectura de datos del cliente de Bob no tendría éxito ya que el cliente ya estaría bloqueado. Cuando Alice guardase los cambios el objeto saldría de la lista y ya Bob puede hacer lo suyo. Para evitar el problema "irse a tomar un café" esa lista habría que escanearla periódicamente y eliminar de ella los que llevan demasiado tiempo (timeout). Esto sería un control pesimista.

c. Hacer que la aplicación añada a cada entidad un campo versión (p.e. un contador). Así, cuando Alice y Bob leen el objeto cliente ambos reciben los datos con la misma versión. Cuando Bob guarda sus cambios el sistema incrementa la versión. Más tarde llegan los datos de Alice, que son de versión anterior (la versión debe ir y venir siempre con los datos, no la puede modificar el usuario, es un campo oculto). En este momento la aplicación compara la versión de los datos que manda Alice con la versión más actual. Si no coinciden se aborta la transacción de aplicación. Alice tendrá que empezar de nuevo. Esto es un control optimista.

d. *Cualquiera de las soluciones puede ser válida. Todo dependerá del contexto de uso, es necesario analizarlo con cuidado. Si es verdad que la probabilidad de colisión es inexistente, y si la hubiera sus efectos no tuvieran consecuencias preocupantes, "No hacer nada" puede ser la opción, aunque no sea lo más elegante (hay muchas aplicaciones por ahí con este potencial problema). Si los efectos de la perdida de actualización nos preocupan, o queremos hacer las cosas bien por defecto, habría que pensar en alguna de las otras dos. La clave estará en la frecuencia de las colisiones y el coste de (re)hacer las modificaciones para el usuario. Con alta frecuencia y coste está claro que el control pesimista es el adecuado. Con baja frecuencia o coste está claro el optimista. En situaciones intermedias hay que hacer una valoración sosegada.

Quitar mi elección

* Las respuestas de esta página serán guardadas automáticamente al pulsar el botón 'Siguiente'

◄ Algunas preguntas interesantes

Ir a...

Cuestionario: Gestión de objetos persistentes ►