TALLER DE PROGRAMACIÓN WEB

Sesión 14

SPRING JDBC - TRANSACCIONES

Computación e Informática

2017 - I

Eric Gustavo Coronel Castillo

gcoronelc.blogspot.com

Encuéntranos en:

INSTITUTO DE EMPRENDEDORES

LOGRO DE LA SESIÓN

Programar transacciones usando Spring JDBC.

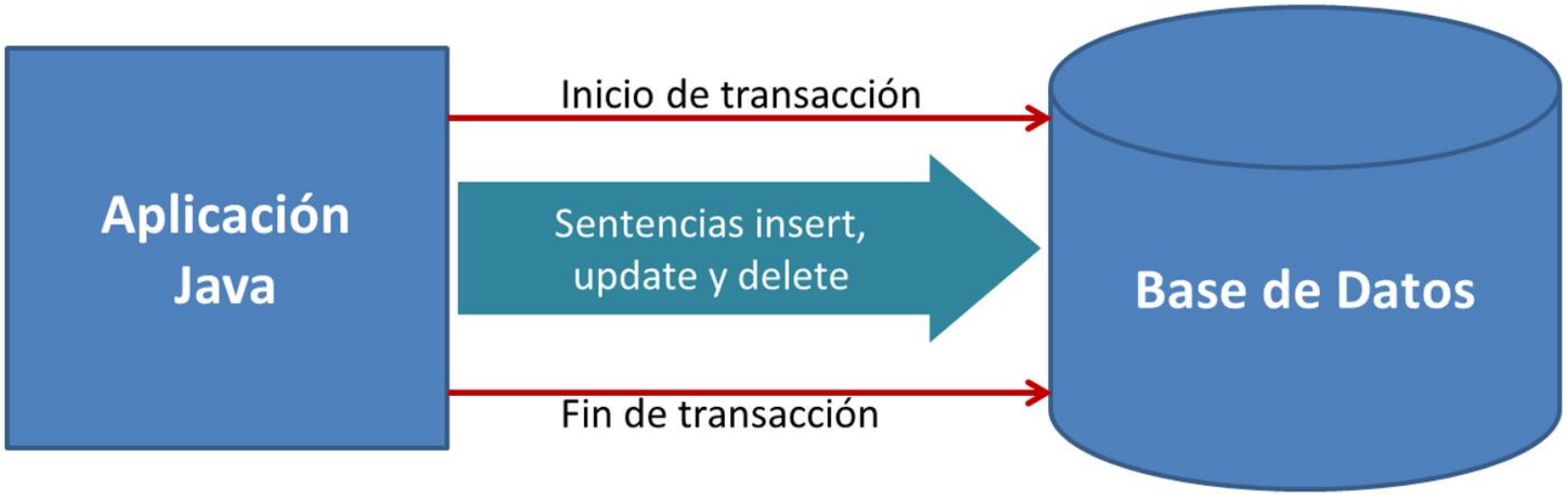
Computación e Informática

2017 - I

Eric Gustavo Coronel Castillo

gcoronelc.blogspot.com

¿QUÉ ES UNA TRANSACCIÓN?



CONFIGURACIÓN

Definiendo el objeto transaccional

```
<bean id="transactionManager" class =
   "org.springframework.jdbc.datasource.DataSourceTransactionManager">
   <property name="dataSource" ref="dataSource" />
</bean>
```

Configurando el uso de anotaciones

```
<tx:annotation-driven transaction-manager="transactionManager" />
```

PROGRAMACIÓN

Anotación @Transactional

```
@Transactional
public void insertar(ClienteBean bean) {

  // Aquí programas la transacción

}
```

PROGRAMACIÓN

Definiendo el nivel de aislamiento

```
@Transactional( isolation=Isolation.<NIVEL> )
```

Los valores que puede tomar son:

- **DEFAULT:** Utiliza el nivel establecido en el SGBD
- **READ_UNCOMMITED:** Este nivel, permite lecturas sobre registros que aún no se han confirmado, esto se conoce como lecturas sucias, puede provocar que tus consultas contengan registros inválidos.
- **READ_COMMITED:** Este nivel, hace que la transacción sólo lea registros que ya estén confirmados.
- **REPEATABLE_READ:** Este nivel no permite lecturas sobre filas que no tengan cambios confirmados, no permite situaciones donde una transacción lee un registro, una segunda transacción altera el registro, y la primera transacción vuelve a leer el registro, obteniendo así diferentes valores la segunda ocasión.
- **SERIALIZABLE:** Este nivel no permite lecturas sucias, lecturas repetibles y lecturas fantasma, la situación donde se hace una consulta, se obtiene una serie de registros, y una transacción inserta un nuevo registro donde se satisface la condición WHERE de la consulta, el nuevo registro sería el fantasma.

PROGRAMACIÓN

Propagación de una transacción

```
@Transactional( propagation=Propagation.<OPCIÓN> )
```

Los valores que puede tomar OPCIÓN son:

- **MANDATORY:** Este atributo obliga a la transacción a ser ejecutada en un contexto transaccional, si es que no existe un contexto transaccional en la ejecución del método Spring genera una excepción de tipo IllegalTransactionStateException.
- **REQUIRED:** Si el método es invocado desde un contexto transaccional, entonces el método será invocado en el mismo contexto transaccional. Si el método no es invocado desde un contexto transaccional, entonces el método creará una nueva transacción e intentará confirmar (commit) la transacción cuando el método termine su ejecución.
- **REQUIRES_NEW:** El método siempre creará una nueva transacción cuando sea invocado y confirmará (commit) la transacción cuando el método termine su ejecución. Si ya existe un contexto transaccional, entonces Spring suspenderá la transacción existente y creará otra transacción, cuando el método termine su ejecución comprometerá la transacción y reanudará la transacción suspendida.

PROGRAMACIÓN

Propagación de una transacción

```
@Transactional( propagation=Propagation.<OPCIÓN> )
```

Los valores que puede tomar OPCIÓN son:

- **NOT_SUPPORTED:** Si el método es ejecutado en un contexto transaccional, entonces este contexto no es propagado a la ejecución del método, por lo que Spring suspenderá el contexto transaccional y lo reanudará cuando el método termine su ejecución.
- **SUPPORTS:** Si ya existe un contexto transaccional, entonces el método será invocado en el mismo contexto transaccional (igual que REQUIRED), si no existe un contexto transaccional entonces no se crea un contexto transaccional (igual que NOT_SUPPORTED)
- **NEVER:** Este atributo obliga que la ejecución del método no sea invocado desde un contexto transaccional, de lo contrario Spring genera una excepción.
- **NESTED:** Se ejecuta dentro de una transacción anidada si un contexto transaccional existe.

PROGRAMACIÓN

Controlando la Cancelación de la Transacción

```
@Transactional(propagation=Propagation.REQUIRED,
    rollbackFor=Exception.class)
public void insertar(ClienteBean bean) throws Exception {

    // Proceso

}
```

PROGRAMACIÓN

Manejando varias excepciones

```
@Transactional(
propagation=Propagation.REQUIRED,
     rollbackForClassName={"Exception"} )
  public void insertar(ClienteBean bean) throws Exception {

    // Proceso

  }
```

INSTITUTO DE
EMPRENDEDORES

USIL