Bases de Datos II

Trabajo Práctico Integrador: Etapa 4

Fecha de Publicación: 16/06
Fecha de Entrega: 29/06

Introducción

Esta cuarta etapa consistirá en proveer un nuevo servicio de persistencia utilizando **Spring Data** para el modelo ya mapeado con Hibernate, en lugar de usar Hibernate directamente como se hizo en el TP1. Para lograr esto, deberá crearse una nueva implementación de `DBliveryService` llamada `SpringDataDBliveryService`, que deberá utilizar repositorios de Spring Data para resolver cualquier acceso a datos requerido.

En este caso se debe crear un repositorio por clase persistente. Los repositorios implementados con Spring Data no son clases sino **interfaces**, y deberán extender `CrudRepository` definiendo los argumentos necesarios: tipo de entidad y tipo de id. Por ejemplo:

```
public interface OrderRepository extends CrudRepository<Order, Long>
```

Las consultas se pueden implementar utilizando métodos query automáticamente implementados por Spring Data (por ejemplo: `findBy<Propiedad>`), o mediante HQL utilizando la annotation @Query (notar que, dado que se trata de una interfaz, los métodos no tienen implementación).

Se han creado dos nuevas clase de tests, `DBliveryServiceTestCase` y `DBliveryStatisticsTestCase` que validan que la implementación de `SpringDataDBliveryService` cumpla con la funcionalidad básica, y un conjunto de consultas, respectivamente. Las consultas ya fueron realizadas en el TP1:

1. **`getAllOrdersMadeByUser(String username)`**
   Obtiene todas las ordenes realizadas por el usuario con username username

2. **`getPendingOrders()`**
   Obtiene el listado de las órdenes pendientes

3. **`getSentOrders()`**
   Obtiene el listado de las órdenes enviadas y no entregadas

4. **`getDeliveredOrdersInPeriod(Date startDate, Date endDate)`**
   Obtiene todas las órdenes entregadas entre dos fechas

5. **`getDeliveredOrdersForUser(String username)`**

Obtiene todas las órdenes entregadas para el cliente con username username

6. **`getProductsOnePrice()`**
   Obtiene los productos que no cambiaron su precio

7. **`getSoldProductsOn(Date day)`**
   Obtiene los productos vendidos en un day

8. **`getMaxWeigth()`**
   Obtiene el producto más pesado

Para configurar SpringData se utiliza la clase **`SpringDataConfiguration`**, que deberán editar con sus credenciales de acceso a la BD y su nro de grupo.

La infraestructura del proyecto provista, incluyendo una versión actualizada del archivo pom.xml con la versión de Spring Data (JPA) que se utilizará, se encuentra en el mismo repositorio del TP previo, **en la rama `springdata`**:

https://github.com/juliangrigera/bdlivery/tree/springdata

Al igual que en la etapa 1, desde la línea de comandos debe obtenerse un build exitoso en donde pasen todos los tests.

Requisitos detallados que deben satisfacerse en esta etapa

1. Escribir todo el código necesario para que la aplicación compile y todos los tests de las clases **`DBliveryServiceTestCase`** y **`DBliveryStatisticsTestCase`** pasen.
2. Correr `mvn clean install` debe resultar en un build exitoso en donde todos los tests pasen.
3. El código tiene que estar subido en el mismo repositorio que la etapa previa, pero en nuevo branch llamado **`practica4`**.
4. La forma de entrega va a ser solamente a través del repositorio privado compartido con el ayudante designado.