«Talento Tech»

Back-End

Java

Clase 12

Buenos Aires Ciudad

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Clase 12: Integración con Base de Datos MySQL

Índice

Objetivos de la Clase

- Comprender el concepto de bases de datos relacionales y la utilidad de MySQL en proyectos empresariales.
- Instalar y configurar MySQL y MySQL Workbench para administrar la base de datos.
- Crear la base de datos necesaria para el proyecto e-commerce de TechLab.
- Conectar la aplicación Spring Boot con MySQL utilizando Spring Data JPA.
- Definir una entidad básica (`Producto`) con anotaciones JPA, generando automáticamente la tabla en la base de datos.
- Sentar las bases para que en la próxima clase implementemos las operaciones CRUD directamente en la base de datos.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Bienvenida a TechLab

Talento Lab

¡Les damos la bienvenida nuevamente a **TechLab**! Hasta ahora hemos creado una API REST que provee datos de productos, pero éstos viven en memoria y se pierden cuando reiniciamos el servidor. Para dar un paso hacia una aplicación más realista y escalable, necesitamos una base de datos que persista la información entre reinicios, actualizaciones y despliegues.

En esta clase, aprenderemos a integrar Spring Boot con una base de datos MySQL. Por el momento, no implementaremos aún las operaciones CRUD directamente contra la base; simplemente prepararemos el entorno, configuraremos las dependencias, crearemos la base de datos y mappearemos nuestras entidades. En la próxima clase, utilizaremos repositorios JPA para realizar operaciones CRUD reales.

Introducción a Bases de Datos Relacionales y MySQL

Una base de datos relacional organiza los datos en tablas con filas y columnas. Esto facilita consultas complejas, integridad de datos y escalabilidad. MySQL es un motor popular, open-source y muy utilizado en la industria.

Los conceptos clave incluyen:

- **Tablas:** Contienen registros de un tipo de entidad (por ejemplo, `producto`).
- **Llaves Primarias (PRIMARY KEY):** Identifican unívocamente cada fila.
- **Relaciones:** Permiten vincular datos entre tablas (por ejemplo, productos y categorías).

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Instalación y Configuración de MySQL y MySQL Workbench

1. **Instalación de MySQL Community Server:**
   Descargar desde https://dev.mysql.com/downloads/ la versión apropiada para tu OS.
   Configurar contraseña para el usuario root.
2. **MySQL Workbench:**
   Herramienta GUI para administrar la base de datos: crear tablas, ejecutar queries, etc.
3. **Inicio del Servidor MySQL:**
   Asegurarse de que el servicio MySQL esté corriendo.

Creación de la Base de Datos para el Proyecto

Desde MySQL Workbench:

```
CREATE DATABASE techlabdb;
USE techlabdb;
```

Hemos creado la base `techlabdb`. Aún no hay tablas. Spring Data JPA se encargará de crearlas cuando ejecutemos la aplicación y definamos nuestras entidades.

Conexión de Spring Boot con MySQL usando Spring Data JPA

Dependencias en el pom.xml

Agregar en `pom.xml` (si aún no lo hiciste):

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Configuración en application.properties

En `src/main/resources/application.properties`:

```
spring.datasource.url=jdbc:mysql://localhost:3306/techlabdb?useSSL=fals
e&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=TU_CONTRASEÑA

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

ddl-auto=update// indicará a Hibernate que genere o actualice las
tablas según las entidades definidas.
```

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Mapeo de Entidades y Relaciones Básicas

Para mapear la clase `Producto` a la tabla `producto`:

```
@Entity
@Table(name = "producto")
public class Producto {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    private String nombre;
    private Double precio;
    private Integer cantidadEnStock;

    // getters y setters
}
```

- `@Entity`: Indica que esta clase se mapeará a una tabla.
- `@Table("producto")`: Nombre de la tabla.
- `@Id`, `@GeneratedValue`: Llave primaria autoincremental.

Relaciones con otras tablas (por ejemplo, `@OneToMany`, `@ManyToOne`) las veremos más adelante. Por ahora, mantenemos una entidad sencilla.

Verificación de la Creación de Tablas

Al ejecutar la aplicación (`mvn spring-boot:run` o `./gradlew bootRun`), Spring Boot conectará con MySQL, y Hibernate creará la tabla `producto` según la entidad.

Podés verificar en MySQL Workbench con:

```
USE techlabdb;
SHOW TABLES;
```

Deberías ver la tabla `producto`.

Por ahora, no haremos CRUD desde la base. Esto quedará para la próxima clase, donde crearemos repositorios y utilizaremos el `ProductoRepository` para guardar, consultar, actualizar y eliminar datos directamente en la base.

Situación Inicial en TechLab

Silvia, la Product Owner, nota que si el servidor se reinicia, se pierden todos los productos que el equipo ha creado a través de la API. Matías y Sabrina saben que la solución es persistir los datos en una base real, como MySQL. Antes de cambiar el servicio para usar la base, deben preparar el entorno: instalar MySQL, configurar la conexión en Spring Boot, y mapear las entidades para que Spring Data JPA pueda gestionar las tablas.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Ejercicios Prácticos

1. **Verificación de Entidad y Tabla:**
    - Objetivo: Confirmar que la entidad `Producto` se mapea correctamente en la base de datos.
    - Tareas:
        1. Asegurate de tener configuradas las dependencias en el `pom.xml` (o `build.gradle`), incluyendo `mysql-connector-j` y `spring-boot-starter-data-jpa`.
        2. Definí la clase `Producto` con las anotaciones `@Entity`, `@Table`, `@Id`, `@GeneratedValue`.
        3. Ejecutá la aplicación y verificá en MySQL Workbench (u otra herramienta) que se crea la tabla `producto` en la base `techlabdb`.

2. **Customización de la Entidad:**
    - Objetivo: Practicar la modificación de entidades y observar el impacto en la base de datos.
    - Tareas:
        1. Agregá un nuevo atributo en la entidad `Producto`, por ejemplo, `String categoria`.
        2. Volvé a ejecutar la aplicación con `spring.jpa.hibernate.ddl-auto=update` habilitado.
        3. Revisá en la base de datos que se haya creado la columna `categoria` en la tabla `producto`.
        4. Comentá por qué es importante mantener sincronizado el modelo de datos (la entidad) con la base.

Materiales y Recursos Adicionales

- **Documentación Oficial de MySQL**:
  https://dev.mysql.com/doc/
  Para profundizar en conceptos de SQL, configuración de usuarios, seguridad y administración general.
- **Spring Boot Reference**:
  https://docs.spring.io/spring-boot/docs/current/reference/html/
  Para entender las propiedades de configuración (por ejemplo, `spring.datasource.*`, `spring.jpa.*`).
- **Spring Data JPA Reference**:
  https://docs.spring.io/spring-data/jpa/docs/current/reference/html/
  Guía para trabajar con repositorios, consultas personalizadas y relaciones.
- **Videos Recomendados**:
  - *"Integración de Spring Boot con MySQL"* (YouTube)
  - *"Configuración de Entidades y JPA"* (YouTube)

Preguntas para Reflexionar

1. **¿Por qué es conveniente definir las entidades con anotaciones como `@Entity` y `@Table` en lugar de escribir sentencias SQL manualmente?**
   - Pista: Pensá en la productividad, mantenibilidad y menor exposición a errores de sintaxis o incompatibilidad.
2. **¿Qué ventajas ofrece `ddl-auto=update` al inicio del proyecto y qué riesgos podría implicar en producción?**
   - Pista: Considerá el beneficio de la autogeneración de tablas y la posibilidad de que en un entorno real, se necesite mayor control sobre los cambios en la base de datos.
3. **¿Cómo ayuda la persistencia en la base de datos a resolver el problema de la pérdida de datos tras reiniciar la aplicación?**
   - Pista: Reflexioná sobre el almacenamiento en memoria vs. almacenamiento en disco y la longevidad de los datos.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Próximos Pasos

En la **Clase 13**, llevaremos este esquema un paso más allá, implementando:

- **Operaciones CRUD (Create, Read, Update, Delete)** directamente sobre la base de datos usando Repositorios JPA (`ProductoRepository`).
- **Refactorización de Servicios y Controladores**, reemplazando la lista en memoria por las operaciones reales a la base.
- **Pruebas de Funcionalidad** con Postman o cURL, confirmando que los cambios se reflejan en la base de datos.

Esto dejará nuestro e-commerce más cerca de un entorno **profesional**, con datos que persisten y se gestionan con facilidad en MySQL. Al finalizar la próxima clase, la API REST podrá crear, consultar, actualizar y eliminar productos de manera confiable y escalable. ¡El proyecto TechLab sigue avanzando a grandes pasos!

Buenos Aires Ciudad