Desarrollo avanzado de aplicaciones II

Tema Nº4:Spring Data JPA I

Indicador de logro Nº4:Conecta su proyecto a una base de datos y define sus objetos en la aplicación web, teniendo en cuenta el modelo de la base de datos.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº4:**

Spring Data JPA I.

**Subtema 4.1:**

Definición de conceptos generales. (Spring Data, JPA, ORM).

**Spring Data:**

Spring Data es un proyecto de SpringSource cuyo propósito es unificar y facilitar el acceso a distintos tipos de tecnologías de persistencia, tanto a bases de datos relacionales

como a las del tipo NoSQL.

Spring ya proporcionaba soporte para JDBC, Hibernate, JPA, JDO o Mybatis, simplificando la implementación de la capa de acceso a datos, unificando la configuración y creando una jerarquía de excepciones común para todas ellas.

Y ahora, Spring Data viene a cubrir el soporte necesario para distintas tecnologías de bases de datos NoSQL y, además, integra las tecnologías de acceso a datos tradicionales, simplificando el trabajo a la hora de crear las implementaciones concretas.

Con cada tipo de tecnología de persistencia los DAOs (Data Access Objects) ofrecen las funcionalidades típicas de un CRUD (Create-Read-Update-Delete ) para objetos de dominio propios, métodos de busqueda, ordenación y paginación. Spring Data proporciona interfaces genéricas para estos aspectos (CrudRepository, PagingAndSortingRepository) e implementaciones específicas para cada tipo de tecnología de persistencia.

A día de hoy, Spring Data proporciona soporte para las siguientes tecnologías de persistencia:

* JPA y JDBC
* Apache Hadoop
* GemFire
* Redis
* MongoDB
* Neo4j
* HBase

En este tutorial vamos a analizar, a través de un ejemplo, el soporte que nos proporciona Spring Data para JPA, haciendo uso de la implementación de referencia, el entityManager de Hibernate.

**JPA**

La persistencia de datos es un medio mediante el cual una aplicación puede recuperar información desde un sistema de almacenamiento no volátil y hacer que esta persista. La persistencia de datos es vital en las aplicaciones empresariales debido al acceso necesario a las bases de datos relacionales. Las aplicaciones desarrolladas para este entorno deben gestionar por su cuenta la persistencia o utilizar soluciones de terceros para manejar las actualizaciones y recuperaciones de las bases de datos con persistencia. JPA (Java™ Persistence API) proporciona un mecanismo para gestionar la persistencia y la correlación relacional de objetos y funciona desde las especificaciones EJB 3.0.

La especificación JPA define la correlación relacional de objetos internamente, en lugar de basarse en implementaciones de correlación específicas del proveedor. JPA se basa en el modelo de programación Java que se aplica a los entornos Java EE (Java Enterprise Edition) pero JPA puede funcionar en un entorno Java SE para probar las funciones de las aplicaciones.

JPA representa una simplificación del modelo de programación de persistencia. La especificación JPA define explícitamente la correlación relacional de objetos, en lugar de basarse en implementaciones de correlación específicas del proveedor. JPA crea un estándar para la importante tarea de la correlación relacional de objetos mediante la utilización de anotaciones o XML para correlacionar objetos con una o más tablas de una base de datos. Para simplificar aún más el modelo de programación de persistencia:

* La API EntityManager puede actualizar, recuperar, eliminar o aplicar la persistencia de objetos de una base de datos.
* La API EntityManager y los metadatos de correlación relacional de objetos manejan la mayor parte de las operaciones de base de datos sin que sea necesario escribir código JDBC o SQL para mantener la persistencia.
* JPA proporciona un lenguaje de consulta, que amplía el lenguaje de consulta EJB independiente, conocido también como JPQL, el cual puede utilizar para recuperar objetos sin grabar consultas SQL específicas en la base de datos con la que está trabajando.

**ORM**

Object-Relational Mapping (ORM) es una técnica de programación para convertir datos entre bases de datos relacionales y lenguajes de programación orientados a objetos como Java, C#, etc.

En la programación, los desarrolladores de Java solían adoptar el enfoque JDBC para la persistencia de datos, que implica escribir y ejecutar sentencias SQL para recuperar/actualizar datos en la base de datos. Este enfoque era popular, pero probablemente no era bueno porque SQL puede ser difícil de leer y escribir, propenso a errores y difícil de depurar.

Para facilitar la programación y reducir la posibilidad de cometer errores, muchos desarrolladores prefieren no ejecutar directamente las sentencias SQL, sino construir un modelo de objetos que refleje la estructura de datos. En tiempo de ejecución, los datos serán recuperados de la base de datos y llenados en el modelo de objetos. Los desarrolladores pueden entonces trabajar completamente con los objetos, sin escribir ninguna sentencia SQL. La técnica para convertir datos entre el modelo de objetos y la base de datos relacional se conoce como mapeo objeto-relacional (mapeo ORM, O/RM y O/R).

**Subtema 4.2:**

Configuración Spring Data JPA y la conexión a una base de datos MySQL.

**Ejemplos:**

Creamos un nuevo proyecto.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Seleccionamos las siguientes dependencias y dar clic a finish.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Word, Sitio web

Descripción generada automáticamente

**Subtema 4.3:**

Anotaciones JPA.

**Ejemplos:**

* @Entity: Marca la clase como una entidad.
* @Table: Define la clase como una tabla de base de datos.
* @Id: Indica que el atributo es clave primaria de la entidad o tabla.
* @GeneratedValue: Indica que los valores del atributo definido será generado automáticamente.

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Subtema 4.4:**

Relación de entidades con anotaciones JPA

**Ejemplos:**

Creamos las siguientes clases:

* Estado:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* Sala

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* Asiento

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Actividad:**

a) CUESTIONARIO TÉCNICO

* ¿Qué es JPA?
* ¿Para que sirve un ORM?
* ¿Qué es Spring Data?
* Cree una entidad con 6 atributos.
* Mapee 3 tablas de su base de datos utilizando las anotaciones de relación.

b) CONCLUSIONES DE LA EXPERIENCIA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_