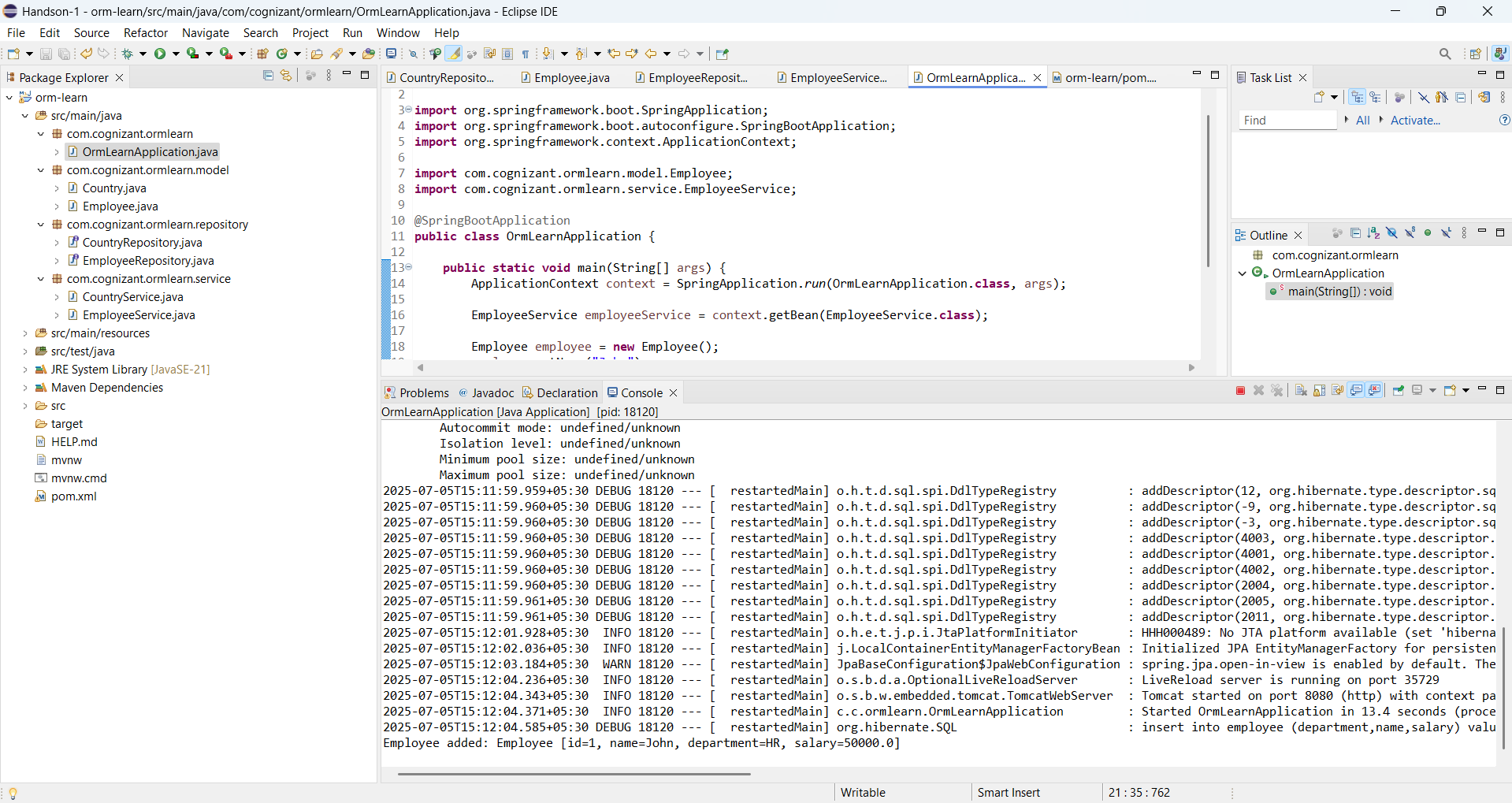
|  |  |  |
| --- | --- | --- |
| **JPA** | **Hibernate** | **Spring Data JPA** |
| Standard annotations (@Entity, @Id, @GeneratedValue, etc.) | Implements JPA  Offers native | Eliminates boilerplate for repository creation |
| Entity lifecycle management | Offers native features beyond JPA like:  Caching (1st & 2nd level) | Spring Data JPA supports interface-based projections |
| Query language: JPQL (Java Persistence Query Language) | Lazy/Eager fetching customization | Provides CrudRepository, JpaRepository, PagingAndSortingRepository |
| Basic CRUD operations | Criteria API | Supports method query derivation (e.g., findByName) |
| Transaction management via EntityManager | Interceptors, Filters, Batch processing | Easy integration with Spring Boot |
| JPA is just a specification — like a set of rules or guidelines | Hibernate is an implementation of JPA — it follows JPA rules but also adds its own features. | Built on top of JPA and typically uses Hibernate under the hood. |
| It defines how Java objects map to database tables. | Offers additional power: caching, HQL, filters, interceptors, etc. | Helps you avoid writing boilerplate code like EntityManager and queries. |
| It uses interfaces like EntityManager but doesn't do the actual work. | Can work with or without JPA. | You define interfaces; Spring auto-creates the repository code. |
| Needs an implementation like Hibernate to function. | Has its own configuration (hibernate.cfg.xml) and annotations. | Supports query methods like findByEmail(), pagination, sorting, etc. |
| Supported by many vendors (Hibernate, Eclipse Link, OpenJPA) | Widely used and mature ORM solution. | Works best with Spring Boot for rapid app development. |

**Hands on 4**



**Hands on 1**

