# Spring Data JPA - Quick Example

Spring Data JPA simplifies database interaction by reducing boilerplate code. In our example, we used Spring Initializer to set up a Spring Boot project with dependencies like Spring Data JPA, Spring DevTools, and MySQL Driver. We created a schema called `ormlearn` and configured our datasource in `application.properties`.  
  
An entity class `Country` was created with `@Entity` and `@Table` annotations. We used `@Id` and `@Column` to map fields to table columns. `CountryRepository` extends `JpaRepository` to give us access to methods like `findAll()`, `save()`, `deleteById()`.  
  
We also created a service class `CountryService` which autowired the repository and exposed methods like `getAllCountries()`. We tested it by loading the Spring context and calling this method from `OrmLearnApplication` using logger to print the retrieved list.

# Difference between JPA, Hibernate, and Spring Data JPA

JPA (Java Persistence API) is a specification that provides a standard for ORM (Object Relational Mapping). It doesn’t provide implementation. Hibernate is an ORM tool that implements JPA. Spring Data JPA is an abstraction over JPA that eliminates boilerplate code and integrates tightly with Spring.  
  
- JPA is just a specification (JSR 338)  
- Hibernate is a powerful ORM framework and an implementation of JPA  
- Spring Data JPA adds a layer of abstraction to reduce complexity and support derived query methods, auditing, pagination, and more  
  
For example, using Hibernate, we write full code to open sessions, begin transactions, etc. With Spring Data JPA, we just autowire a repository and call `save()` or `findById()`.