1. SQL Data

Spring framework cung cấp khả năng làm việc với SQL Database

+ From direct JDBC access using JdbcTemplate

+ ORM technologies such as Hibernate

+ Creating Repository implementations directly from interfaces and using conventions to generate queries from your method names.

Connection Pool can use:

* HikariCP (default)
* Tomcat pooling Datasource
* Commons DBCP2
* Oracle UCP & OracleDataSource
* Spring Framework’s SimpleDriverDataSource
* H2 JdbcDataSource
* PostgreSQL PGSimpleDataSource
* C3P0

Config using application.property

Config using java bean have class DataSourceBuilder support

Spring’s JdbcTemplate and NamedParameterJdbcTemplate classes are auto-configured, and you can @Autowire inject into own bean to run

* JPA va Spring data JPA:

The Java Persistence API is a standard technology “map” objects to relational databases. The spring-boot-starter-data-jpa POM provides a quick way to get started. It provides the following key dependencies:

* Hibernate: One of the most popular JPA implementations.
* Spring Data JPA: Helps you to implement JPA-based repositories.
* Spring ORM: Core ORM support from the Spring Framework.

All Entity bean (@Entity, @Embeddable, or @MappedSuperclass are considered) in package have location < SpringbootApp.class default scanning by one annotated with @EnableAutoConfiguration or @SpringBootApplication in class main.

You can customize entity scanning locations by using the @EntityScan annotation

Tips: In Entity bean class, constructor no-args

// no-args constructor required by JPA spec

// this one is protected since it should not be used directly

**Spring Data JPA Repositories**

Truy vấn dữ liệu trong spring data jpa có thể sử dụng:

* + Kho interface repository default declare avaiable of Spring data jpa (have method default and also can mehtod custom). JPA Queries automatic genarate
  + Using @Query on method to custom query

Spring Data JPA repositories support three different modes of bootstrapping: default, deferred, and lazy

#### Spring Data Envers Repositories

If [Spring Data Envers](https://spring.io/projects/spring-data-envers) is available, JPA repositories are auto-configured to support typical Envers queries, To use Spring Data Envers, make sure your repository extends from RevisionRepository

Code generation using JOOQ

Genarate entity Pojo :

+ Add plugin into file pom.xml

<build>  
 <plugins>

<plugin>  
 <groupId>org.jooq</groupId>  
 <artifactId>jooq-codegen-maven</artifactId>  
 <executions>  
 <execution>  
 <id>jooq-codegen</id>  
 <phase>generate-sources</phase>  
 <goals>  
 <goal>generate</goal>  
 </goals>  
 <configuration>  
 *<!--Insert your DB configuration-->* <jdbc>  
 <driver>com.mysql.cj.jdbc.Driver</driver>  
 <url>jdbc:mysql://localhost:3306/mydb</url>  
 <user>root</user>  
 <password>20150601</password>  
 </jdbc>  
 <generator>  
 <database>  
 <name>org.jooq.meta.mysql.MySQLDatabase</name>  
 <includes>.\*</includes>  
 <excludes></excludes>  
 <inputSchema>mydb</inputSchema>  
 </database>  
 <generate>  
 <pojos>true</pojos>  
 <pojosEqualsAndHashCode>  
 true  
 </pojosEqualsAndHashCode>  
 <javaTimeTypes>true</javaTimeTypes>  
 <fluentSetters>true</fluentSetters>  
 </generate>  
 <target>  
 <packageName>  
 entity  
 </packageName>  
 <directory>  
 target/generated-sources/jooq  
 </directory>  
 </target>  
 </generator>  
 </configuration>  
 </execution>  
 </executions>  
 </plugin>  
 </plugins>  
</build>

+ Add dependency:

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-jooq</artifactId>  
</dependency>

+ Config DataSource using java class config or file application.properties

Finaly Running : mvn clean install (in terminal)

### Using R2DBC

 R2DBC’s io.r2dbc.spi.Connection provides a standard method of working with non-blocking database connections. Connections are provided by using a ConnectionFactory, similar to a DataSource with jdbc

A DatabaseClient bean is auto-configured, and you can @Autowire it directly into your own beans,

## Working with NoSQL Technologies

Spring Data provides additional projects that help you access a variety of NoSQL technologies, including:

* [MongoDB](https://spring.io/projects/spring-data-mongodb)
* [Neo4J](https://spring.io/projects/spring-data-neo4j)
* [Elasticsearch](https://spring.io/projects/spring-data-elasticsearch)
* [Redis](https://spring.io/projects/spring-data-redis)
* [GemFire](https://spring.io/projects/spring-data-gemfire) or [Geode](https://spring.io/projects/spring-data-geode)
* [Cassandra](https://spring.io/projects/spring-data-cassandra)
* [Couchbase](https://spring.io/projects/spring-data-couchbase)
* [LDAP](https://spring.io/projects/spring-data-ldap)

### Using the ApplicationRunner or CommandLineRunner

Dùng 2 interface này để là gì đó sau khi app run, implement 1 trong 2 để sử dụng

Nếu có nhiều beans CommandLineRunner or ApplicationRunner được khởi tạo, phải dùng @Order

### Configuring Random Values

Ex. \*.properties: my.secret=${random.value}

my.number=${random.int}

my.bignumber=${random.long}

Using @PropertySource(….) using @Value(“${my.secret}”)

Bind property value from .properties to bean:

+ Binding qua thuộc tính

Declare: @Component  
@PropertySource("classpath:my-application.properties")  
@ConfigurationProperties(prefix = "my.service") trên class bean

+ Bindding qua contructor

Declare: @Component  
@PropertySource("classpath:my-application.properties")

@ConstructorBinding   
@ConfigurationProperties(prefix = "my.service") trên class bean