Spring_Boot with JPA(Hibernate) to ORACLE:-

1. while creating the spring boot starter project with eclipse or spring initializer we have to add below dependencies.

i. spring boot starter web dependency (it will download all the web suported jars)
ii. add **devtools dependency** (auto restart the server when ever you modify anything in the classpath)
i**ii.actuator** dependency(for Monitoring purpose)
iv. spring boot starter data jpa dependency (it is include by default tomcat connection pool and h2 database and derby database ---)

to use external databases like oracle and other connection pool like hikari

<!-- Oracle JDBC driver -->

```
<dependency>
  <groupId>com.oracle</groupId>
  <artifactId>ojdbc7</artifactId>
  <version>12.1.0</version>
</dependency>   // by default it might not available,
```

use the below command to setup to your local maven(add to local maven repository ojdbc)

Note: mvn install:install-file -Dfile=C:/app/oracle/product/12.1.0/dbhome_1/jdbc/lib/ojdbc7.jar -DgroupId=com.oracle -DartifactId=ojdbc7 -Dversion=12.1.0 -Dpackaging=jar

<!-- HikariCP connection pool -->

```
<dependency>
  <groupId>com.zaxxer</groupId>
  <artifactId>HikariCP</artifactId>
</dependency>
```

Note: before run the project please add below points to your application.properties file

Application.properties

```
spring.datasource.driver-class-name=oracle.jdbc.driver.OracleDriver
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:orcl
spring.datasource.username=system
spring.datasource.password=oracle

spring.datasource.hikari.connection-timeout=20000
spring.datasource.hikari.maximum-pool-size=3000
```

Note:-- connection pool

* whenever we will add starter-data by default jdbc connection pool available if you don't want default connection pool, if you want to use external connection pool like hikari, you just remove the default connection pool and add respective connection pool dependency . ex: see in pom.xml

pom.xml::

```
		<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.2.4.RELEASE</version>
		<relativePath /> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.example</groupId>
	<artifactId>DNS_IP</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>DNS_IP</name>
	<description>Demo project for Spring Boot</description>

	<properties>
		<java.version>1.8</java.version>
	</properties>

	<dependencies>

		<!-- Oracle JDBC driver -->
		<dependency>
			<groupId>com.oracle</groupId>
			<artifactId>ojdbc7</artifactId>
			<version>12.1.0</version>
		</dependency>
		<!-- HikariCP connection pool -->
		<dependency>
			<groupId>com.zaxxer</groupId>
			<artifactId>HikariCP</artifactId>
		</dependency>


		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-actuator</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
			<exclusions>
				<exclusion>
					<groupId>org.apache.tomcat</groupId>
					<artifactId>tomcat-jdbc</artifactId>
				</exclusion>
			</exclusions>
		</dependency>
```

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

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-devtools</artifactId>
			<scope>runtime</scope>
			<optional>true</optional>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
			<exclusions>
				<exclusion>
					<groupId>org.junit.vintage</groupId>
					<artifactId>junit-vintage-engine</artifactId>
				</exclusion>
			</exclusions>
		</dependency>

		<dependency>
			<groupId>io.springfox</groupId>
			<artifactId>springfox-swagger2</artifactId>
			<version>2.9.2</version>
		</dependency>
		<dependency>
			<groupId>io.springfox</groupId>
			<artifactId>springfox-swagger-ui</artifactId>
			<version>2.6.1</version>
		</dependency>


	</dependencies>

	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
			</plugin>
		</plugins>
	</build>

</project>
```

Entity class: -- refer to your database tables

Note:-- I have a table in my database, having 4 columns like below

```
@Entity
@Table(name = "logindetails")
public class Login {

@Id
@Column(name = "id")
@GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "CUST_SEQ")
@SequenceGenerator(sequenceName = "customer_seq", allocationSize = 1, name =
"CUST_SEQ")
private Long id;

@Column(name = "name")
private String name;

@Column(name = "password")
private String pwd;

@Column(name = "dob")
private String date;


// setter and getter methods

}
```

Repository interface::-- create one repository it extends the Repository either directly or indirectly

Note:--- JPA provide some default Repositories we can use it and override it.

```
@Repository
public interface UserRepository extends CrudRepository<Login, String> {

}
```

Service class:-- create one service class which should petch data from the repository.

```
@Service
public class LoginDao {

	@Autowired
	public UserRepository userRepository;

	public List<Login> loginDetails() {

		List<Login> loginList = new ArrayList<>();
		Iterable<Login> findAll = userRepository.findAll();
		for (Login login : findAll) {
			loginList.add(login);

		}
		return loginList;
```

```
	}
}
```

RestPoint:-- create one controller class (restcontroller) for access

```
@RestController
public class DataBaseConfig {

	@Autowired
	LoginDao loginDao;

	@RequestMapping("/login")
	public  List<Login> guru() {

		return loginDao.loginDetails();
	}
}
```

Main Class: --

```
@EnableJpaRepositories(basePackages = "com.example.demo")
@SpringBootApplication()
public class DnsIpApplication {

	public static void main(String[] args) {
		SpringApplication.run(DnsIpApplication.class, args);
	}
   }
```