

Week-3 Hands On Spring Core and Maven

Exercise 1: Configuring a Basic Spring Application

```
<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>
  <groupId>com.library</groupId>
  <artifactId>LibraryManagement</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.34</version>
    </dependency>
  </dependencies>
</project>
```

```
package com.library.repository;

import java.util.Arrays;
import java.util.List;

public class BookRepository {
    public List<String> getAllBooks() {
        return Arrays.asList("RD Sharma", "RS Agarwal", "MBD
Guide");
    }
}

package com.library.service;

import com.library.repository.BookRepository;
import java.util.List;

public class BookService {
```

```

    private BookRepository bookRepository;

    public void setBookRepository(BookRepository
bookRepository) {
        this.bookRepository = bookRepository;
    }

    public void displayBooks() {
        List<String> books = bookRepository.getAllBooks();
        System.out.println("Available Books:");
        books.forEach(System.out::println);
    }
}

```

```

<?xml_version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-
beans.xsd">

    <bean id="bookRepository"
class="com.library.repository.BookRepository"/>

    <bean id="bookService"
class="com.library.service.BookService">
        <property name="bookRepository" ref="bookRepository"/>
    </bean>

</beans>

```

```

package com.library;
import com.library.service.BookService;
import org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApplicationCon
text;

public class LibraryApp {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

        BookService bookService =
context.getBean(BookService.class);
    }
}

```

```

        bookService.displayBooks(); // AOP will wrap
this
    }
}

```

OUTPUT:

```

<terminated> LibraryApp [Java Application] C:\Users\admin\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7
Available Books:
The Notebook
WhiteNights
A Walk to Remember

```

Exercise 2: Implementing Dependency Injection

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- Define BookRepository bean -->
    <bean id="bookRepository"
class="com.library.repository.BookRepository" />

    <!-- Define BookService bean with DI -->
    <bean id="bookService"
class="com.library.service.BookService">
        <property name="bookRepository" ref="bookRepository"
/>
    </bean>

</beans>

```

```

package com.library.service;

import com.library.repository.BookRepository;
import java.util.List;

public class BookService {
    private BookRepository bookRepository;

    public void setBookRepository(BookRepository
bookRepository) {
        this.bookRepository = bookRepository;
    }

    public void displayBooks() {
        List<String> books = bookRepository.getAllBooks();
        System.out.println("Available Books:");
        books.forEach(System.out::println);
    }
}

```

```

package com.library.repository;

import java.util.Arrays;
import java.util.List;

public class BookRepository {
    public List<String> getAllBooks() {
        return Arrays.asList("RD Sharma", "RS Agarwal", "MBD
Guide");
    }
}

```

```

package com.library;

import com.library.service.BookService;
import org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApplicationCon
text;

public class LibraryApp {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

        BookService service = context.getBean("bookService",
BookService.class);

        service.displayBooks();
    }
}

```

```
}  
}
```

OUTPUT:

```
<terminated> LibraryApp [Java Application] C:\Users\admin\p  
Available Books:  
RD Sharma  
RS Agarwal  
MBD Guide
```

Exercise 4: Creating and Configuring a Maven Project

//Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.

```
<dependencies>  
  <dependency>  
    <groupId>org.springframework</groupId>  
    <artifactId>spring-context</artifactId>  
    <version>5.3.34</version>  
  </dependency>  
  
  <dependency>  
    <groupId>org.springframework</groupId>  
    <artifactId>spring-aop</artifactId>  
    <version>5.3.34</version>  
  </dependency>  
  
  <dependency>  
    <groupId>org.aspectj</groupId>  
    <artifactId>aspectjweaver</artifactId>  
    <version>1.9.21</version>  
  </dependency>  
</dependencies>
```

//Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

```
<build>  
  <plugins>
```

```

    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.11.0</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>
  </plugins>
</build>

```

Exercise 5: Configuring the Spring IoC Container

```

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {
    private BookRepository bookRepository;

    public void setBookRepository(BookRepository
bookRepository) {
        this.bookRepository = bookRepository;
    }

    public void displayBooks() {
        for (String book : bookRepository.getAllBooks()) {
            System.out.println(book);
        }
    }
}

```

//Defining beans for BookService and BookReository

```

<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-
beans.xsd">

```

```

    <bean id="bookRepository"
class="com.library.repository.BookRepository"/>

    <bean id="bookService"
class="com.library.service.BookService">
    <property name="bookRepository" ref="bookRepository"/>
    </bean>

</beans>

```

```

package com.library;

import com.library.service.BookService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryApp {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

        BookService service = context.getBean("bookService",
BookService.class);
        service.displayBooks();
    }
}

```

OUTPUT:

```

terminated> LibraryApp [Java Application] C:\Users\admin\p2\poof\plugins\org.eclipse.just\openjdk.hotspot.jre.fu
The Notebook
WhiteNights
A Walk to Remember

```

Exercise 7: Implementing Constructor and Setter Injection

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="
           http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/context
           http://www.springframework.org/schema/context/spring-
           context.xsd">

    <context:component-scan base-package="com.library" />

    <bean id="bookRepository"
class="com.library.repository.BookRepository" />

    <bean id="bookService"
class="com.library.service.BookService">
        <constructor-arg value="LibraryService" />
        <property name="bookRepository" ref="bookRepository"
/>
    </bean>
</beans>
```

```
package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

    private String serviceName;
    private BookRepository bookRepository;

    public BookService(String serviceName) {
        this.serviceName = serviceName;
        System.out.println("BookService created with service
name: " + serviceName);
    }

    public void setBookRepository(BookRepository
bookRepository) {
```



```

        this.bookRepository = bookRepository;
    }

    public void displayBooks() {
        System.out.println("Service: " + serviceName);
        bookRepository.getAllBooks().forEach(System.out::println);
    }
}

```

```

package com.library.repository;

import org.springframework.stereotype.Repository;
import java.util.List;
import java.util.Arrays;

@Repository
public class BookRepository {

    public List<String> getAllBooks() {
        return Arrays.asList("RD Sharma", "A Walk To Remember");
    }
}

```

```

package com.library;

import com.library.service.BookService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryApp {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

        BookService bookService = context.getBean(BookService.class);
        bookService.displayBooks();
    }
}

```

OUTPUT:

```
<terminated> LibraryApp [Java Application] C:\Users\admin\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32
BookService created with service name: LibraryService
Service: LibraryService
RD Sharma
A Walk To Remember
```

9.Creating a Spring Boot Application

XML Files:

```
<dependencies>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
  <groupId>com.h2database</groupId>
  <artifactId>h2</artifactId>
  <scope>runtime</scope>
</dependency>
</dependencies>
```

```
package com.library.repository;
import com.library.entity.Book;
import org.springframework.data.jpa.repository.JpaRepository;
public interface BookRepository extends JpaRepository<Book, Long>
{
}

package com.library.controller;
import com.library.entity.Book;
import com.library.repository.BookRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import java.util.List;
@RestController
@RequestMapping("/books")
public class BookController {
    @Autowired
```

```

    private BookRepository bookRepository;
    @GetMapping
    public List<Book> getAllBooks() {
        return bookRepository.findAll();
    }
    @GetMapping("/{id}")
    public Book getBookById(@PathVariable Long id) {
        return bookRepository.findById(id).orElse(null);
    }
    @PostMapping
    public Book createBook(@RequestBody Book book) {
        return bookRepository.save(book);
    }
    @PutMapping("/{id}")
    public Book updateBook(@PathVariable Long id, @RequestBody
    Book bookDetails) {
        Book book = bookRepository.findById(id).orElse(null);
        if (book != null) {
            book.setTitle(bookDetails.getTitle());
            book.setAuthor(bookDetails.getAuthor());
            return bookRepository.save(book);
        }
        return null;
    }
    @DeleteMapping("/{id}")
    public String deleteBook(@PathVariable Long id) {
        bookRepository.deleteById(id);
        return "Book deleted with id: " + id;
    }
}

```

```

package com.library;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class LibraryManagementApplication {
    public static void main(String[] args) {

SpringApplication.run(LibraryManagementApplication.class,
args);
    }
}

```

```

package com.library.entity;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;

```

```
import jakarta.persistence.Id;
@Entity
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String title;
    private String author;
    public Book() {}
    public Book(String title, String author) {
        this.title = title;
        this. author = author;
    }
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }

    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }

    public String getAuthor() { return author; }
    public void setAuthor(String author) { this.author = author; }
}
```

spring-data-jpa-handson

Spring Data JPA - Quick Example

```
spring.application.name=orm-learn
# DATABASE
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=

# HIBERNATE
spring.jpa.hibernate.ddl-auto=update
spring.jpa.defer-datasource-initialization=true

# LOGGING & DEBUGGING
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
logging.level.org.hibernate.SQL=debug
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=trace

# H2 CONSOLE
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console
```

```
package com.cognizant.orm_learn;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.cognizant.orm_learn.Country;

@Repository
public interface CountryRepository extends
JpaRepository<Country, String> {
}
```

```
package com.cognizant.orm_learn;

import com.cognizant.orm_learn.Country;
import com.cognizant.orm_learn.CountryRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```

import jakarta.transaction.Transactional;
import java.util.List;

@Service
public class CountryService {

    @Autowired
    private CountryRepository countryRepository;

    @Transactional
    public List<Country> getAllCountries() {
        return countryRepository.findAll();
    }
}

```

```

package com.cognizant.orm_learn;

import com.cognizant.orm_learn.Country;
import com.cognizant.orm_learn.CountryService;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;

import java.util.List;

@SpringBootApplication
public class OrmLearnApplication {

    private static CountryService service;

    private static final Logger Log =
        LoggerFactory.getLogger(OrmLearnApplication.class);

    public static void main(String[] args) {
        ApplicationContext context =
            SpringApplication.run(OrmLearnApplication.class, args);
        Log.info("Application started successfully.");

        service = context.getBean(CountryService.class);

        showCountries(); // Just a test to fetch all data
    }
}

```

```

private static void showCountries() {
    Log.info("Fetching countries...");

    List<Country> countryList = service.getAllCountries();

    for (Country c : countryList) {
        System.out.println(c); // just printing for now
    }

    Log.info("Done fetching countries.");
}
}

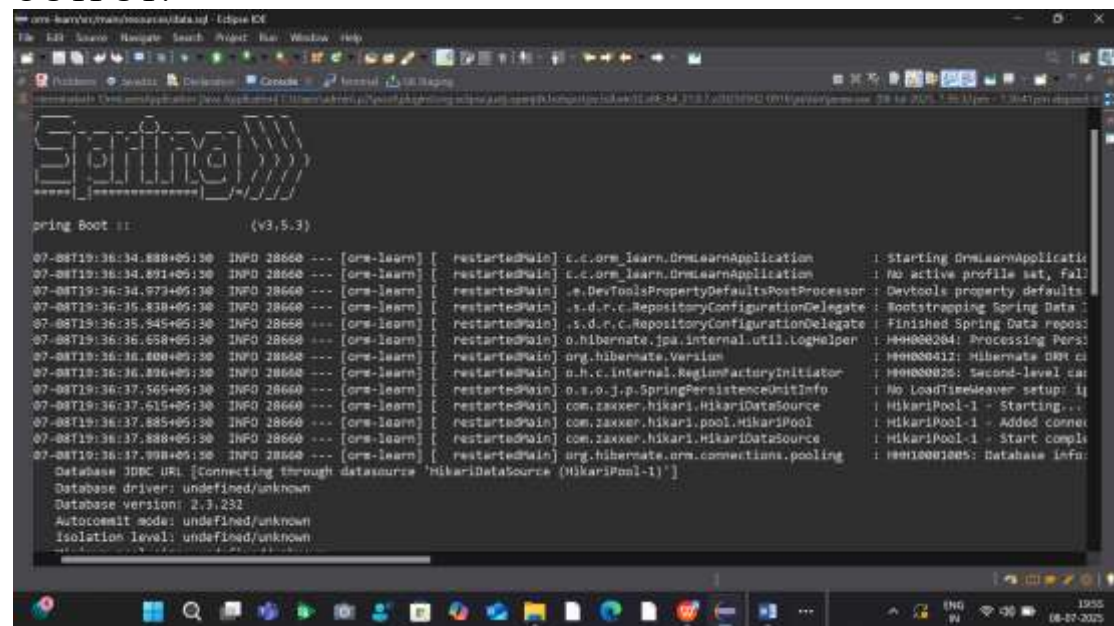
```

```

insert into country (co_code, co_name) values ('IN', 'India');
insert into country (co_code, co_name) values ('US', 'United
States');

```

OUTPUT:



```

Spring Boot :: (v3.5.3)
07-08T19:36:34.888+05:30 INFO 28660 --- [orm-learn] [ restarted@Main] c.c.orm_learn.OrmLearnApplication : Starting OrmLearnApplication
07-08T19:36:34.891+05:30 INFO 28660 --- [orm-learn] [ restarted@Main] c.c.orm_learn.OrmLearnApplication : No active profile set, fall
07-08T19:36:34.973+05:30 INFO 28660 --- [orm-learn] [ restarted@Main] e.DevToolsPropertyDefaultsPostProcessor : DevTools property defaults
07-08T19:36:35.839+05:30 INFO 28660 --- [orm-learn] [ restarted@Main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data
07-08T19:36:35.845+05:30 INFO 28660 --- [orm-learn] [ restarted@Main] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repos
07-08T19:36:36.658+05:30 INFO 28660 --- [orm-learn] [ restarted@Main] o.hibernate.jpa.internal.util.LogHelper : HHH0000284: Processing Persi
07-08T19:36:36.808+05:30 INFO 28660 --- [orm-learn] [ restarted@Main] org.hibernate.Version : HHH0000412: Hibernate ORM v
07-08T19:36:36.896+05:30 INFO 28660 --- [orm-learn] [ restarted@Main] o.h.c.internal.RegionFactoryInitiator : HHH0000026: Second-level ca
07-08T19:36:37.565+05:30 INFO 28660 --- [orm-learn] [ restarted@Main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup; ty
07-08T19:36:37.615+05:30 INFO 28660 --- [orm-learn] [ restarted@Main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
07-08T19:36:37.885+05:30 INFO 28660 --- [orm-learn] [ restarted@Main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connec
07-08T19:36:37.888+05:30 INFO 28660 --- [orm-learn] [ restarted@Main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start compl
07-08T19:36:37.998+05:30 INFO 28660 --- [orm-learn] [ restarted@Main] org.hibernate.orm.connections.pooling : HHH10001005: Database info:
Database JDBC URL: [Connecting through datasource 'HikariDataSource (HikariPool-1)']
Database driver: undefined/unknown
Database version: 2.3.231
Autocommit mode: undefined/unknown
Isolation level: undefined/unknown

```

```

Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown
2025-07-08T19:36:39.228+05:30 INFO 28660 --- [orm-learn] [ restartedMain] o.h.w.e.t.j.p.l.JtaPlatformInitiator : HHH000489: No JTA pla
2025-07-08T19:36:39.288+05:30 DEBUG 28660 --- [orm-learn] [ restartedMain] org.hibernate.SQL
create table country (
  co_code varchar(255) not null,
  co_name varchar(255),
  primary key (co_code)
)
Hibernate:
create table country (
  co_code varchar(255) not null,
  co_name varchar(255),
  primary key (co_code)
)
2025-07-08T19:36:39.299+05:30 INFO 28660 --- [orm-learn] [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA Entity
2025-07-08T19:36:40.122+05:30 INFO 28660 --- [orm-learn] [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is
2025-07-08T19:36:40.160+05:30 INFO 28660 --- [orm-learn] [ restartedMain] c.c.orm.learn.OmLearnApplication : Started OmLearnAppli
2025-07-08T19:36:40.170+05:30 INFO 28660 --- [orm-learn] [ restartedMain] c.c.orm.learn.OmLearnApplication : Application started s
2025-07-08T19:36:40.170+05:30 INFO 28660 --- [orm-learn] [ restartedMain] c.c.orm.learn.OmLearnApplication : Fetching countries...
2025-07-08T19:36:40.478+05:30 DEBUG 28660 --- [orm-learn] [ restartedMain] org.hibernate.SQL
select
  c1_0.co_code,
  c1_0.co_name
from
  country c1_0
Hibernate:
select
  c1_0.co_code,
  c1_0.co_name
from
  country c1_0
Country(code='IN', name='India')
Country(code='US', name='United States')
2025-07-08T19:36:40.547+05:30 INFO 28660 --- [orm-learn] [ restartedMain] c.c.orm.learn.OmLearnApplication : Done fetching countri
2025-07-08T19:36:40.556+05:30 INFO 28660 --- [orm-learn] [ onShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityMan
2025-07-08T19:36:40.566+05:30 WARN 28660 --- [orm-learn] [ onShutdownHook] o.s.b.f.support.DisposableBeanAdapter : Invocation of destroy
2025-07-08T19:36:40.568+05:30 INFO 28660 --- [orm-learn] [ onShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdo
2025-07-08T19:36:40.568+05:30 INFO 28660 --- [orm-learn] [ onShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdo

```

Difference between JPA, Hibernate and Spring Data JPA

```

package com.cognizant.orm_learn;

import jakarta.persistence.*;

@Entity
@Table(name = "employee")
public class Employee {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)

```



```

private int id;

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

@Column(name = "salary")
private double salary;

// Constructors
public Employee() {
}

public Employee(String name, double salary) {
    this.name = name;
    this.salary = salary;
}

// Getters & Setters
public int getId() {
    return id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public double getSalary() {
    return salary;
}

public void setSalary(double salary) {
    this.salary = salary;
}
}

```

```

package com.cognizant.orm_learn;
import com.cognizant.orm_learn.Employee;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface EmployeeRepository extends
JpaRepository<Employee, Integer> {
}

```



```
2025-07-10T07:57:13.205+05:30 INFO 31556 --- [orm-learn] [
restartedMain] c.c.orm_learn.OrmLearnApplication      :
Starting OrmLearnApplication using Java 21.0.7 with PID 31556
(C:\Users\admin\Downloads\orm-learn\orm-learn\target\classes
started by admin in C:\Users\admin\Downloads\orm-learn\orm-
learn)
2025-07-10T07:57:13.207+05:30 INFO 31556 --- [orm-learn] [
restartedMain] c.c.orm_learn.OrmLearnApplication      : No
active profile set, falling back to 1 default profile:
"default"
2025-07-10T07:57:13.287+05:30 INFO 31556 --- [orm-learn] [
restartedMain] .e.DevToolsPropertyDefaultsPostProcessor :
Devtools property defaults active! Set 'spring.devtools.add-
properties' to 'false' to disable
2025-07-10T07:57:14.217+05:30 INFO 31556 --- [orm-learn] [
restartedMain] .s.d.r.c.RepositoryConfigurationDelegate :
Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2025-07-10T07:57:14.327+05:30 INFO 31556 --- [orm-learn] [
restartedMain] .s.d.r.c.RepositoryConfigurationDelegate :
Finished Spring Data repository scanning in 93 ms. Found 2 JPA
repository interfaces.
2025-07-10T07:57:14.998+05:30 INFO 31556 --- [orm-learn] [
restartedMain] o.hibernate.jpa.internal.util.LogHelper  :
HHH000204: Processing PersistenceUnitInfo [name: default]
2025-07-10T07:57:15.105+05:30 INFO 31556 --- [orm-learn] [
restartedMain] org.hibernate.Version                  :
HHH000412: Hibernate ORM core version 6.6.18.Final
2025-07-10T07:57:15.172+05:30 INFO 31556 --- [orm-learn] [
restartedMain] o.h.c.internal.RegionFactoryInitiator        :
HHH000026: Second-level cache disabled
2025-07-10T07:57:15.746+05:30 INFO 31556 --- [orm-learn] [
restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo              : No
LoadTimeWeaver setup: ignoring JPA class transformer
2025-07-10T07:57:15.784+05:30 INFO 31556 --- [orm-learn] [
restartedMain] com.zaxxer.hikari.HikariDataSource                :
HikariPool-1 - Starting...
2025-07-10T07:57:16.030+05:30 INFO 31556 --- [orm-learn] [
restartedMain] com.zaxxer.hikari.pool.HikariPool                  :
HikariPool-1 - Added connection conn0: url=jdbc:h2:mem:testdb
user=SA
2025-07-10T07:57:16.032+05:30 INFO 31556 --- [orm-learn] [
restartedMain] com.zaxxer.hikari.HikariDataSource                :
HikariPool-1 - Start completed.
2025-07-10T07:57:16.127+05:30 INFO 31556 --- [orm-learn] [
restartedMain] org.hibernate.orm.connections.pooling              :
HHH10001005: Database info:
      Database JDBC URL [Connecting through datasource
'HikariDataSource (HikariPool-1)']
      Database driver: undefined/unknown
```

```
Database version: 2.3.232
Autocommit mode: undefined/unknown
Isolation level: undefined/unknown
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown
2025-07-10T07:57:17.380+05:30 INFO 31556 --- [orm-learn] [
restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator      :
HHH000489: No JTA platform available (set
'hibernate.transaction.jta.platform' to enable JTA platform
integration)
2025-07-10T07:57:17.383+05:30 INFO 31556 --- [orm-learn] [
restartedMain] j.LocalContainerEntityManagerFactoryBean :
Initialized JPA EntityManagerFactory for persistence unit
'default'
2025-07-10T07:57:18.370+05:30 INFO 31556 --- [orm-learn] [
restartedMain] o.s.b.d.a.OptionalLiveReloadServer      :
LiveReload server is running on port 35729
2025-07-10T07:57:18.403+05:30 INFO 31556 --- [orm-learn] [
restartedMain] c.c.orm_learn.OrmLearnApplication        :
Started OrmLearnApplication in 5.886 seconds (process running
for 6.593)
2025-07-10T07:57:18.492+05:30 DEBUG 31556 --- [orm-learn] [
restartedMain] org.hibernate.SQL                          :
        insert
        into
            employee
            (name, salary, id)
        values
            (?, ?, default)
Hibernate:
        insert
        into
            employee
            (name, salary, id)
        values
            (?, ?, default)
2025-07-10T07:57:18.643+05:30 DEBUG 31556 --- [orm-learn] [
restartedMain] org.hibernate.SQL                          :
        insert
        into
            employee
            (name, salary, id)
        values
            (?, ?, default)
Hibernate:
        insert
        into
            employee
            (name, salary, id)
```

```

        values
            (?, ?, default)
All Employees:
2025-07-10T07:57:18.822+05:30 DEBUG 31556 --- [orm-learn] [
restartedMain] org.hibernate.SQL
        select
            e1_0.id,
            e1_0.name,
            e1_0.salary
        from
            employee e1_0
Hibernate:
        select
            e1_0.id,
            e1_0.name,
            e1_0.salary
        from
            employee e1_0
1 | Alice | 65000.0
2 | Bob | 72000.0
101 | Alice | 65000.0
102 | Bob | 72000.0
2025-07-10T07:57:18.864+05:30 INFO 31556 --- [orm-learn]
[ionShutdownHook] j.LocalContainerEntityManagerFactoryBean :
Closing JPA EntityManagerFactory for persistence unit
'default'
2025-07-10T07:57:18.873+05:30 WARN 31556 --- [orm-learn]
[ionShutdownHook] o.s.b.f.support.DisposableBeanAdapter :
Invocation of destroy method failed on bean with name
'inMemoryDatabaseShutdownExecutor':
org.h2.jdbc.JdbcSQLException: Database
is already closed (to disable automatic closing at VM
shutdown, add ";DB_CLOSE_ON_EXIT=FALSE" to the db URL) [90121-
232]
2025-07-10T07:57:18.874+05:30 INFO 31556 --- [orm-learn]
[ionShutdownHook] com.zaxxer.hikari.HikariDataSource :
HikariPool-1 - Shutdown initiated...
2025-07-10T07:57:18.877+05:30 INFO 31556 --- [orm-learn]
[ionShutdownHook] com.zaxxer.hikari.HikariDataSource :
HikariPool-1 - Shutdown completed.

```

JPA (Java Persistence API)	Hibernate	Spring Data JPA
A <i>specification</i> (standard) to define how to manage relational data in Java.	A <i>framework/tool</i> that implements the JPA specification.	A <i>Spring-based abstraction</i> that sits on top of JPA (and Hibernate).

JPA (Java Persistence API)	Hibernate	Spring Data JPA
JPA is just a set of guidelines and interfaces.	Hibernate is a real tool that does the job.	It delegates to Hibernate (or another JPA provider) internally.
Java EE (Jakarta EE) / Oracle	Red Hat (Hibernate ORM Project)	Spring Framework (Pivotal / VMware)
A lot — need to write EntityManager, handle transactions manually.	Less than JDBC, but still need Session, Transaction mgmt.	Very little — uses JpaRepository, @Transactional, etc.
Low-level configuration using EntityManager	Similar, but uses Session API	High-level: just create interface, Spring does the rest!
Manual or via container	Manual using Transaction object	Handled automatically with @Transactional
javax.persistence	org.hibernate	org.springframework.data.jpa
em.persist(obj) with EntityManager	session.save(obj) with transaction mgmt	employeeRepository.save(obj) automatically handled

Implement services for managing country

```

spring.application.name=orm-learn
# DATABASE
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=

#HIBERNATE
spring.jpa.defer-datasource-initialization=true

# LOGGING & DEBUGGING
spring.jpa.show-sql=true

```

```

spring.jpa.properties.hibernate.format_sql=true
logging.level.org.hibernate.SQL=debug
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=trace

# H2 CONSOLE
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console
spring.datasource.initialization-mode=always
# \u2705 Enable schema initialization
spring.sql.init.mode=always

# \u2705 Optional for development \u2014 lets Hibernate create
the schema
spring.jpa.hibernate.ddl-auto=none

```

```

package com.cognizant.orm_learn;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name = "country")
public class Country {
    @Id
    @Column(name = "co_code")
    private String code;

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

    public String getCode() {
        return code;
    }

    public void setCode(String code) {
        this.code = code;
    }

    public String getName() {
        return name;
    }
}

```

```

        public void setName(String name) {
            this.name = name;
        }
    }
}

```

```

package com.cognizant.orm_learn;

import com.cognizant.orm_learn.Country;
import com.cognizant.orm_learn.CountryRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;

@Service
public class CountryService {

    @Autowired
    private CountryRepository countryRepository;

    public Country findCountryByCode(String code) {
        return countryRepository.findById(code).orElse(null);
    }

    public void addCountry(Country country) {
        countryRepository.save(country);
    }

    public void updateCountry(String code, String newName) {
        Country country =
countryRepository.findById(code).orElse(null);
        if (country != null) {
            country.setName(newName);
            countryRepository.save(country);
        }
    }

    public void deleteCountry(String code) {
        countryRepository.deleteById(code);
    }

    public List<Country> findCountriesByPartialName(String
partial) {
        return
countryRepository.findByNameContaining(partial);
    }
}

```



```

package com.cognizant.orm_learn;

import com.cognizant.orm_learn.Country;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.List;

public interface CountryRepository extends
JpaRepository<Country, String> {
    List<Country> findByNameContaining(String partialName);
}

```

```

package com.cognizant.orm_learn;

import com.cognizant.orm_learn.Country;
import com.cognizant.orm_learn.CountryService;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner
{

    @Autowired
    private CountryService countryService;

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

    @Override
    public void run(String... args) {
        System.out.println("Fetching country by code: IN");
        Country india =
countryService.findCountryByCode("IN");
        System.out.println(india.getName());

        System.out.println("Adding new country...");
        Country test = new Country();
        test.setCode("XX");
        test.setName("Testland");
        countryService.addCountry(test);

        System.out.println("Updating country XX...");
    }
}

```

```

countryService.updateCountry("XX", "Test Republic");

System.out.println("Searching partial 'Re'...");

countryService.findCountriesByPartialName("Re").forEach(c ->
System.out.println(c.getName()));

System.out.println("Deleting country XX...");
countryService.deleteCountry("XX");
}
}

```

The screenshot shows an IDE window with the following content:

Spring Boot (v3.5.3)

```

2025-07-09T11:02:43.411+05:30 INFO 26748 --- [orm-learn] [ restartedMain] c.c.orm_learn.OreLearnApplication : Starting OreLearnAppli
2025-07-09T11:02:43.417+05:30 INFO 26748 --- [orm-learn] [ restartedMain] c.c.orm_learn.OreLearnApplication : No active profile set
2025-07-09T11:02:43.512+05:30 INFO 26748 --- [orm-learn] [ restartedMain] e.DevToolsPropertyDefaultsPostProcessor : Devtools property def
2025-07-09T11:02:44.387+05:30 INFO 26748 --- [orm-learn] [ restartedMain] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring I
2025-07-09T11:02:44.483+05:30 INFO 26748 --- [orm-learn] [ restartedMain] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data :
2025-07-09T11:02:45.075+05:30 INFO 26748 --- [orm-learn] [ restartedMain] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing
2025-07-09T11:02:45.178+05:30 INFO 26748 --- [orm-learn] [ restartedMain] org.hibernate.Version : HHH000041:2: Hibernate I
2025-07-09T11:02:45.234+05:30 INFO 26748 --- [orm-learn] [ restartedMain] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-lev
2025-07-09T11:02:45.757+05:30 INFO 26748 --- [orm-learn] [ restartedMain] o.s.i.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver set
2025-07-09T11:02:46.186+05:30 INFO 26748 --- [orm-learn] [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starti
2025-07-09T11:02:46.114+05:30 INFO 26748 --- [orm-learn] [ restartedMain] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added :
2025-07-09T11:02:46.118+05:30 INFO 26748 --- [orm-learn] [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start :
2025-07-09T11:02:46.218+05:30 INFO 26748 --- [orm-learn] [ restartedMain] org.hibernate.orm.connections.pooling : HHH000100: Database
Database JDBC URL [Connecting through
Database driver: undefined/unknown
Database version: 1.2.232
Autocommit mode: undefined/unknown

```

Maximus pool size: undefined/unknown

```

2025-07-09T11:02:47.357+05:30 INFO 26748 --- [orm-learn] [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA pla
2025-07-09T11:02:47.402+05:30 INFO 26748 --- [orm-learn] [ restartedMain] j.localContainerEntityManagerFactoryBean : Initialized JPA Entiti
2025-07-09T11:02:48.046+05:30 INFO 26748 --- [orm-learn] [ restartedMain] o.s.b.s.a.OptionalLiveReloadServer : LiveReload server is
2025-07-09T11:02:48.087+05:30 INFO 26748 --- [orm-learn] [ restartedMain] c.c.orm_learn.OreLearnApplication : Started OreLearnAppli

```

Fetching country by code: IN

```

2025-07-09T11:02:48.613+05:30 DEBUG 26748 --- [orm-learn] [ restartedMain] org.hibernate.SQL
select
  c1_0.co_code,
  c1_0.co_name
from
  country c1_0
where
  c1_0.co_code=?
Hibernate:
select
  c1_0.co_code,
  c1_0.co_name
from
  country c1_0
where
  c1_0.co_code=?
India
Adding new country...
2025-07-09T11:02:48.809+05:30 DEBUG 26748 --- [orm-learn] [ restartedMain] org.hibernate.SQL
select
  c1_0.co_code,

```

The top screenshot shows an IDE with the following SQL query and log output:

```

c1_0.co_name
from
  country c1_0
where
  c1_0.co_code=?
Hibernate:
select
  c1_0.co_code,
  c1_0.co_name
from
  country c1_0
where
  c1_0.co_code=?
2025-07-09T11:02:48.875+05:30 DEBUG 26748 --- [ore-learn] [ restartedMain] org.hibernate.SQL
insert
into
  country
(co_name, co_code)
values
  (?, ?)
Hibernate:
insert
into
  country
(co_name, co_code)
values
  (?, ?)

```

The bottom screenshot shows an IDE with the following SQL query and log output:

```

Updating country XX...
2025-07-09T11:02:48.902+05:30 DEBUG 26748 --- [ore-learn] [ restartedMain] org.hibernate.SQL
select
  c1_0.co_code,
  c1_0.co_name
from
  country c1_0
where
  c1_0.co_code=?
Hibernate:
select
  c1_0.co_code,
  c1_0.co_name
from
  country c1_0
where
  c1_0.co_code=?
2025-07-09T11:02:48.905+05:30 DEBUG 26748 --- [ore-learn] [ restartedMain] org.hibernate.SQL
select
  c1_0.co_code,
  c1_0.co_name
from
  country c1_0
where
  c1_0.co_code=?
Hibernate:
select

```

```

c1_0.co_code,
c1_0.co_name
from
country c1_0
where
c1_0.co_code=?
2025-07-09T11:02:48.917+05:30 DEBUG 26748 --- [orm-learn] [ restartedMain] org.hibernate.SQL
update
country
set
co_name=?
where
co_code=?
Hibernate:
update
country
set
co_name=?
where
co_code=?
Searching partial 'Re'...
2025-07-09T11:02:49.130+05:30 DEBUG 26748 --- [orm-learn] [ restartedMain] org.hibernate.SQL
select
c1_0.co_code,
c1_0.co_name
from
country c1_0
where
c1_0.co_name like ? escape '\'
Hibernate:
select
c1_0.co_code,
c1_0.co_name
from
country c1_0
where
c1_0.co_name like ? escape '\'
Central African Republic
Congo, the Democratic Republic of the
Czech Republic
Dominican Republic
Iran, Islamic Republic
Democratic Peoples Republic of Korea
Republic of Korea
Lao Peoples Democratic Republic
Macedonia, the Former Yugoslav Republic
Syrian Arab Republic
Tanzania, United Republic of
Venezuela, Bolivarian Republic of
Test Republic
Deleting country XX...
2025-07-09T11:02:49.146+05:30 DEBUG 26748 --- [orm-learn] [ restartedMain] org.hibernate.SQL
select
c1_0.co_code,
```

The screenshot shows an IDE with a SQL query in the editor and its execution logs in the console.

SQL Query:

```

c1_0.co_name
from
  country c1_0
where
  c1_0.co_code=?
Hibernate:
select
  c1_0.co_code,
  c1_0.co_name
from
  country c1_0
where
  c1_0.co_code=?
2025-07-09T11:02:49.163+05:30 DEBUG 26748 --- [orm-learn] [ restartedMain] org.hibernate.SQL
delete
from
  country
where
  co_code=?
Hibernate:
delete
from
  country
where
  co_code=?
2025-07-09T11:02:49.196+05:30 INFO 26748 --- [orm-learn] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory
2025-07-09T11:02:49.203+05:30 WARN 26748 --- [orm-learn] [ionShutdownHook] o.s.b.f.support.DisposableBeanAdapter : Invocation of destroy

```

Console Logs:

```

2025-07-09T11:02:49.203+05:30 WARN 26748 --- [orm-learn] [ionShutdownHook] o.s.b.f.support.DisposableBeanAdapter : Invocation of destroy
2025-07-09T11:02:49.203+05:30 INFO 26748 --- [orm-learn] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdo
2025-07-09T11:02:49.203+05:30 INFO 26748 --- [orm-learn] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdo

```

```

    }

    public void setCoName(String coName) {
        this.coName = coName;
    }

    @Override
    public String toString() {
        return "Country{" +
            "code='" + coCode + '\'' +
            ", name='" + coName + '\'' +
            '}';
    }
}

```

```

package com.cognizant.orm_learn;

import com.cognizant.orm_learn.Country;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.List;

public interface CountryRepository extends
JpaRepository<Country, String> {
    List<Country> findByCoNameContaining(String coName);
}

```

```

package com.cognizant.orm_learn;

import com.cognizant.orm_learn.Country;
import com.cognizant.orm_learn.CountryRepository;
import com.cognizant.orm_learn.CountryNotFoundException;
import jakarta.transaction.Transactional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class CountryService {

    @Autowired
    private CountryRepository countryRepository;

    @Transactional
    public Country findCountryByCode(String countryCode)
throws CountryNotFoundException {
        Optional<Country> result =
countryRepository.findById(countryCode);

```

```

        if (!result.isPresent()) {
            throw new CountryNotFoundException("Country not
found for code: " + countryCode);
        }
        return result.get();
    }
}

```

```

package com.cognizant.orm_learn;

import com.cognizant.orm_learn.Country;
import com.cognizant.orm_learn.CountryService;
import com.cognizant.orm_learn.CountryNotFoundException;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;

@SpringBootApplication
public class OrmLearnApplication {

    private static final Logger LOGGER =
LoggerFactory.getLogger(OrmLearnApplication.class);

    private static CountryService countryService;

    public static void main(String[] args) {
        ApplicationContext context =
SpringApplication.run(OrmLearnApplication.class, args);
        countryService =
context.getBean(CountryService.class);

        getAllCountriesTest();
    }

    private static void getAllCountriesTest() {
        LOGGER.info("Start");

        try {
            Country country =
countryService.findCountryByCode("IN");
            LOGGER.debug("Country: {}", country);
        } catch (CountryNotFoundException e) {
            LOGGER.error("Exception: {}", e.getMessage());
        }
    }
}

```



```
LOGGER.info("End");
```

```

=====
:: Spring Boot ::                (v3.5.3)

2025-07-09T17:46:06.541+05:30 INFO 21952 --- [orm-learn] [ restartedMain] c.c.orm_learn.OmLearnApplication : Starting OmLearn
2025-07-09T17:46:06.544+05:30 INFO 21952 --- [orm-learn] [ restartedMain] c.c.orm_learn.OmLearnApplication : No active profil
2025-07-09T17:46:06.644+05:30 INFO 21952 --- [orm-learn] [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools proper
2025-07-09T17:46:07.661+05:30 INFO 21952 --- [orm-learn] [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Sp
2025-07-09T17:46:07.774+05:30 INFO 21952 --- [orm-learn] [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring
2025-07-09T17:46:08.605+05:30 INFO 21952 --- [orm-learn] [ restartedMain] o.hibernate.jpa.internal.util.LogHelper : HHH000284: Proce
2025-07-09T17:46:08.762+05:30 INFO 21952 --- [orm-learn] [ restartedMain] org.hibernate.Version : HHH000412: Hiber
2025-07-09T17:46:08.858+05:30 INFO 21952 --- [orm-learn] [ restartedMain] o.h.c.internal.RegionFactoryInitiator : HHH000826: Seco
2025-07-09T17:46:09.596+05:30 INFO 21952 --- [orm-learn] [ restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo : No loadTimeWeave
2025-07-09T17:46:09.647+05:30 INFO 21952 --- [orm-learn] [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - S
2025-07-09T17:46:09.968+05:30 INFO 21952 --- [orm-learn] [ restartedMain] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - A
2025-07-09T17:46:09.965+05:30 INFO 21952 --- [orm-learn] [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - S
2025-07-09T17:46:10.089+05:30 INFO 21952 --- [orm-learn] [ restartedMain] org.hibernate.orm.connections.pooling : HHH000160: Dat
Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
Database driver: undefined/unknown
Database version: 2.3.232
Autocommit mode: undefined/unknown

Hibernate: 
    select
      ci_o.co_code,
      ci_o.co_name
    from
      country ci_o
    where
      ci_o.co_code=?
Hibernate: 
    select
      ci_o.co_code,
      ci_o.co_name
    from
      country ci_o
    where
      ci_o.co_code=?
2025-07-09T17:46:13.146+05:30 INFO 21952 --- [orm-learn] [ restartedMain] c.c.orm_learn.OmLearnApplication : End
2025-07-09T17:46:13.165+05:30 INFO 21952 --- [orm-learn] [ onShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityMan
2025-07-09T17:46:13.177+05:30 WARN 21952 --- [orm-learn] [ onShutdownHook] o.s.b.f.support.DisposableBeanAdapter : Invocation of destroy
2025-07-09T17:46:13.177+05:30 WARN 21952 --- [orm-learn] [ onShutdownHook] o.s.b.f.support.DisposableBeanAdapter : Invocation of destroy
2025-07-09T17:46:13.177+05:30 INFO 21952 --- [orm-learn] [ onShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdo
2025-07-09T17:46:13.181+05:30 INFO 21952 --- [orm-learn] [ onShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdo

```


Add a new country

```
package com.cognizant.orm_learn;

import com.cognizant.orm_learn.Country;
import com.cognizant.orm_learn.CountryRepository;
import com.cognizant.orm_learn.CountryNotFoundException;
import jakarta.transaction.Transactional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class CountryService {

    @Autowired
    private CountryRepository countryRepository;

    @Transactional
    public Country findCountryByCode(String countryCode)
throws CountryNotFoundException {
        Optional<Country> result =
countryRepository.findById(countryCode);
        if (!result.isPresent()) {
            throw new CountryNotFoundException("Country not
found for code: " + countryCode);
        }
        return result.get();
    }

    @Transactional
    public void addCountry(Country country) {
        countryRepository.save(country);
    }
}
```

```
package com.cognizant.orm_learn;

import com.cognizant.orm_learn.Country;
import com.cognizant.orm_learn.CountryService;
import com.cognizant.orm_learn.CountryNotFoundException;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
```

```

import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;

@SpringBootApplication
public class OrmLearnApplication {

    private static final Logger LOGGER =
LoggerFactory.getLogger(OrmLearnApplication.class);
    private static CountryService countryService;

    public static void main(String[] args) {
        ApplicationContext context =
SpringApplication.run(OrmLearnApplication.class, args);
        countryService =
context.getBean(CountryService.class);

        getAllCountriesTest();    // Test fetching by code
        testAddCountry();        // Test adding new country
    }

    private static void getAllCountriesTest() {
        LOGGER.info("Start");
        try {
            Country country =
countryService.findCountryByCode("IN");
            LOGGER.debug("Country: {}", country);
        } catch (CountryNotFoundException e) {
            LOGGER.error("Country not found!", e);
        }
        LOGGER.info("End");
    }

    private static void testAddCountry() {
        LOGGER.info("Start - Add Country");

        Country newCountry = new Country();
        newCountry.setCoCode("NP");
        newCountry.setCoName("Nepal");

        countryService.addCountry(newCountry);

        try {
            Country country =
countryService.findCountryByCode("NP");
            LOGGER.debug("Added Country: {}", country);
        } catch (CountryNotFoundException e) {
            LOGGER.error("Country not found after
adding!", e);

```

```

    }

    LOGGER.info("Add Country");

}
}

```

OUTPUT:

```

:: Spring Boot :: (v3.5.3)

2025-07-09T17:56:25.588+05:30 INFO 20116 --- [orm-learn] [ restartedMain] c.c.orm_learn.OrnLearnApplication : Starting OrnLearnAppl
2025-07-09T17:56:25.592+05:30 INFO 20116 --- [orm-learn] [ restartedMain] c.c.orm_learn.OrnLearnApplication : No active profile set
2025-07-09T17:56:25.671+05:30 INFO 20116 --- [orm-learn] [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property def
2025-07-09T17:56:26.693+05:30 INFO 20116 --- [orm-learn] [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring
2025-07-09T17:56:26.795+05:30 INFO 20116 --- [orm-learn] [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data
2025-07-09T17:56:27.512+05:30 INFO 20116 --- [orm-learn] [ restartedMain] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing
2025-07-09T17:56:27.657+05:30 INFO 20116 --- [orm-learn] [ restartedMain] org.hibernate.Version : HHH000412: Hibernate
2025-07-09T17:56:27.745+05:30 INFO 20116 --- [orm-learn] [ restartedMain] o.h.c.internal.RegionFactoryInitiator : HHH00026: Second-lev
2025-07-09T17:56:28.271+05:30 INFO 20116 --- [orm-learn] [ restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver set
2025-07-09T17:56:28.317+05:30 INFO 20116 --- [orm-learn] [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starti
2025-07-09T17:56:28.588+05:30 INFO 20116 --- [orm-learn] [ restartedMain] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added
2025-07-09T17:56:28.590+05:30 INFO 20116 --- [orm-learn] [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start
2025-07-09T17:56:28.697+05:30 INFO 20116 --- [orm-learn] [ restartedMain] org.hibernate.orm.connections.pooling : HHH10001005: Database
Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
Database driver: undefined/unknown
Database version: 2.3.232
Autocommit mode: undefined/unknown

    c1_0.co_name
  from
    country c1_0
  where
    c1_0.co_code=?
2025-07-09T17:56:31.238+05:30 DEBUG 20116 --- [orm-learn] [ restartedMain] org.hibernate.SQL :
select
  c1_0.co_code,
  c1_0.co_name
from
  country c1_0
where
  c1_0.co_code=?
Hibernate:
select
  c1_0.co_code,
  c1_0.co_name
from
  country c1_0
where
  c1_0.co_code=?
2025-07-09T17:56:31.240+05:30 INFO 20116 --- [orm-learn] [ restartedMain] c.c.orm_learn.OrnLearnApplication : End - Add Country
2025-07-09T17:56:31.255+05:30 INFO 20116 --- [orm-learn] [ ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManager
2025-07-09T17:56:31.262+05:30 WARN 20116 --- [orm-learn] [ ionShutdownHook] o.s.b.f.support.DisposableBeanAdapter : Invocation of destroy
2025-07-09T17:56:31.263+05:30 INFO 20116 --- [orm-learn] [ ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdo
2025-07-09T17:56:31.266+05:30 INFO 20116 --- [orm-learn] [ ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdo

```

```

Isolation level: undefined/unknown
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown
2025-07-09T17:56:29.801+05:30 INFO 20116 --- [orm-learn] [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA plat
2025-07-09T17:56:29.807+05:30 INFO 20116 --- [orm-learn] [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA Entit
2025-07-09T17:56:30.894+05:30 INFO 20116 --- [orm-learn] [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is
2025-07-09T17:56:30.935+05:30 INFO 20116 --- [orm-learn] [ restartedMain] c.c.orm_learn.OrmLearnApplication : Started OrmLearnAppli
2025-07-09T17:56:30.943+05:30 INFO 20116 --- [orm-learn] [ restartedMain] c.c.orm_learn.OrmLearnApplication : Start
2025-07-09T17:56:31.034+05:30 DEBUG 20116 --- [orm-learn] [ restartedMain] org.hibernate.SQL :
select
  c1_0.co_code,
  c1_0.co_name
from
  country c1_0
where
  c1_0.co_code=?
hibernate:
select
  c1_0.co_code,
  c1_0.co_name
from
  country c1_0
where
  c1_0.co_code=?
2025-07-09T17:56:31.221+05:30 INFO 20116 --- [orm-learn] [ restartedMain] c.c.orm_learn.OrmLearnApplication : End
2025-07-09T17:56:31.222+05:30 INFO 20116 --- [orm-learn] [ restartedMain] c.c.orm_learn.OrmLearnApplication : Start - Add Country
2025-07-09T17:56:31.232+05:30 DEBUG 20116 --- [orm-learn] [ restartedMain] org.hibernate.SQL :

```

Demonstrate implementation of Query method features of Spring data JPA

Write queries on country table using Query Methods

```
package com.cognizant.orm_learn;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
import jakarta.persistence.Column;

@Entity
@Table(name = "country")
public class Country {

    @Id
    @Column(name = "co_code")
    private String code;

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

    public String getCode() {
        return code;
    }

    public void setCode(String code) {
        this.code = code;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

```
package com.cognizant.orm_learn;
```

```

import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.cognizant.orm_learn.Country;

@Repository
public interface CountryRepository extends
JpaRepository<Country, String> {
    List<Country> findByNameContaining(String keyword); //
works if field is 'name'
    List<Country> findByNameContainingOrderByNameAsc(String
keyword);
    List<Country> findByNameStartingWith(String prefix);
}

```

```

package com.cognizant.orm_learn;

import com.cognizant.orm_learn.Country;
import com.cognizant.orm_learn.CountryRepository;
import com.cognizant.orm_learn.CountryNotFoundException;
import jakarta.transaction.Transactional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class CountryService {

    @Autowired
    private CountryRepository countryRepository;

    @Transactional
    public Country findCountryByCode(String countryCode)
throws CountryNotFoundException {
        Optional<Country> result =
countryRepository.findById(countryCode);
        if (!result.isPresent()) {
            throw new CountryNotFoundException("Country not
found for code: " + countryCode);
        }
        return result.get();
    }

    @Transactional
    public void addCountry(Country country) {
        countryRepository.save(country);
    }
}

```

```

package com.cognizant.orm_learn;

import com.cognizant.orm_learn.Country;
import com.cognizant.orm_learn.CountryService;
import com.cognizant.orm_learn.CountryNotFoundException;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner
{

    @Autowired
    private CountryRepository countryRepository;

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

    @Override
    public void run(String... args) throws Exception {

        // Task 1: Search countries by keyword in name
        System.out.println("\n[1] Countries containing
'ou':");
        List<Country> countriesWithOu =
countryRepository.findByNameContaining("ou");
        countriesWithOu.forEach(c ->
System.out.println(c.getCode() + " - " + c.getName()));

        // Task 2: Search and sort countries by keyword in
name
        System.out.println("\n[2] Countries containing 'ou'
(sorted ascending):");
        List<Country> countriesSorted =
countryRepository.findByNameContainingOrderByNameAsc("ou");
        countriesSorted.forEach(c ->
System.out.println(c.getCode() + " - " + c.getName()));

        // Task 3: Countries starting with 'Z'

```

```
System.out.println("\n[3] Countries starting with  
'Z':");  
  
List<Country> countriesStartingWithZ =  
countryRepository.findByNameStartingWith("Z");  
countriesStartingWithZ.forEach(c ->  
System.out.println(c.getCode() + " - " + c.getName()));  
}  
}
```

OUTPUT:

[illegible]

[illegible]

```

        c1_0.co_name
    from
        country c1_0
    where
        c1_0.co_name like ? escape '\\'
Hibernate:
    select
        c1_0.co_code,
        c1_0.co_name
    from
        country c1_0
    where
        c1_0.co_name like ? escape '\\'
BV - Bouvet Island
DJ - Djibouti
TF - French Southern Territories
GP - Guadeloupe
LU - Luxembourg
ZA - South Africa
GS - South Georgia and the South Sandwich Islands
SS - South Sudan

[2] Countries containing 'ou' (sorted ascending):
2025-07-09T19:32:36.884+05:30 DEBUG 32660 --- [orm-learn] [
restartedMain] org.hibernate.SQL
:
    select
        c1_0.co_code,
        c1_0.co_name
    from
        country c1_0
    where
        c1_0.co_name like ? escape '\\'
    order by
        c1_0.co_name
Hibernate:
    select
        c1_0.co_code,
        c1_0.co_name
    from
        country c1_0
    where
        c1_0.co_name like ? escape '\\'
    order by
        c1_0.co_name
BV - Bouvet Island
DJ - Djibouti
TF - French Southern Territories
GP - Guadeloupe
LU - Luxembourg

```

```

ZA - South Africa
GS - South Georgia and the South Sandwich Islands
SS - South Sudan

[3] Countries starting with 'Z':
2025-07-09T19:32:36.891+05:30 DEBUG 32660 --- [orm-learn] [
restartedMain] org.hibernate.SQL                               :
    select
        c1_0.co_code,
        c1_0.co_name
    from
        country c1_0
    where
        c1_0.co_name like ? escape '\'
Hibernate:
    select
        c1_0.co_code,
        c1_0.co_name
    from
        country c1_0
    where
        c1_0.co_name like ? escape '\'
ZM - Zambia
ZW - Zimbabwe
2025-07-09T19:32:36.909+05:30 INFO 32660 --- [orm-learn]
[ionShutdownHook] j.LocalContainerEntityManagerFactoryBean :
Closing JPA EntityManagerFactory for persistence unit
'default'
2025-07-09T19:32:36.916+05:30 WARN 32660 --- [orm-learn]
[ionShutdownHook] o.s.b.f.support.DisposableBeanAdapter      :
Invocation of destroy method failed on bean with name
'inMemoryDatabaseShutdownExecutor':
org.h2.jdbc.JdbcSQLNonTransientConnectionException: Database
is already closed (to disable automatic closing at VM
shutdown, add ";DB_CLOSE_ON_EXIT=FALSE" to the db URL) [90121-
232]
2025-07-09T19:32:36.917+05:30 INFO 32660 --- [orm-learn]
[ionShutdownHook] com.zaxxer.hikari.HikariDataSource          :
HikariPool-1 - Shutdown initiated...
2025-07-09T19:32:36.919+05:30 INFO 32660 --- [orm-learn]
[ionShutdownHook] com.zaxxer.hikari.HikariDataSource          :
HikariPool-1 - Shutdown completed.

```

Write queries on stock table using Query Methods

```
CREATE TABLE IF NOT EXISTS `stock` (  
  `st_id` INT NOT NULL AUTO_INCREMENT,  
  `st_code` varchar(10),  
  `st_date` date,  
  `st_open` numeric(10,2),  
  `st_close` numeric(10,2),  
  `st_volume` numeric,  
  PRIMARY KEY (`st_id`)  
);
```

```
INSERT INTO stock (st_code, st_date, st_open, st_close,  
st_volume) VALUES  
( 'FB', '2019-09-03', 184.00, 182.39, 9779400),  
( 'FB', '2019-09-04', 184.65, 187.14, 11308000),  
( 'GOOGL', '2019-04-22', 1236.67, 1253.76, 954200),  
( 'GOOGL', '2019-04-23', 1256.64, 1270.59, 1593400),  
( 'NFLX', '2018-12-21', 263.83, 246.39, 21397600),  
( 'NFLX', '2018-12-24', 242.00, 233.88, 9547600),  
( 'NFLX', '2018-12-26', 233.92, 253.67, 14402700),  
( 'FB', '2019-01-31', 165.60, 166.69, 77233600);
```

```
package com.cognizant.orm_learn;  
  
import jakarta.persistence.*;  
import java.time.LocalDate;  
  
@Entity  
@Table(name = "stock")  
public class Stock {  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private int stId;  
  
    private String stCode;  
    private LocalDate stDate;  
    private Double stOpen;  
    private Double stClose;  
    private Long stVolume;  
  
    // Getters and Setters  
    public int getStId() { return stId; }  
    public void setStId(int stId) { this.stId = stId; }  
  
    public String getStCode() { return stCode; }
```

```

    public void setStCode(String stCode) { this.stCode =
stCode; }

    public LocalDate getStDate() { return stDate; }
    public void setStDate(LocalDate stDate) { this.stDate =
stDate; }

    public Double getStOpen() { return stOpen; }
    public void setStOpen(Double stOpen) { this.stOpen =
stOpen; }

    public Double getStClose() { return stClose; }
    public void setStClose(Double stClose) { this.stClose =
stClose; }
}

```

```

package com.cognizant.orm_learn;

import com.cognizant.orm_learn.Stock;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

import java.time.LocalDate;
import java.util.List;

public interface StockRepository extends JpaRepository<Stock,
Integer> {

    List<Stock> findByStCodeAndStDateBetween(String stCode,
LocalDate startDate, LocalDate endDate);

    List<Stock> findByStCodeAndStCloseGreaterThan(String
stCode, double closePrice);

    List<Stock> findTop3ByOrderByStVolumeDesc();

    List<Stock> findTop3ByStCodeOrderByStCloseAsc(String
stCode);
}

```

```

package com.cognizant.orm_learn;

import com.cognizant.orm_learn.Stock;
import com.cognizant.orm_learn.StockRepository;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;

```

```

import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;

import java.time.LocalDate;
import java.util.List;

@SpringBootApplication
public class OrmLearnApplication {
    private static final Logger LOGGER =
LoggerFactory.getLogger(OrmLearnApplication.class);
    private static StockRepository stockRepository;

    public static void main(String[] args) {
        ApplicationContext context =
SpringApplication.run(OrmLearnApplication.class, args);
        stockRepository =
context.getBean(StockRepository.class);

        testFacebookSeptember2019();
        testGooglePriceGreaterThan1250();
        testTop3Volume();
        testNetflixLowest3();
    }

    private static void testFacebookSeptember2019() {
        LOGGER.info("Fetching Facebook stocks in Sep 2019");
        LocalDate start = LocalDate.of(2019, 9, 1);
        LocalDate end = LocalDate.of(2019, 9, 30);
        List<Stock> stocks =
stockRepository.findByStCodeAndStDateBetween("FB", start,
end);
        stocks.forEach(stock ->
LOGGER.debug(stock.toString()));
    }

    private static void testGooglePriceGreaterThan1250() {
        LOGGER.info("Fetching Google stocks > 1250");
        List<Stock> stocks =
stockRepository.findByStCodeAndStCloseGreaterThan("GOOGL",
1250);
        stocks.forEach(stock ->
LOGGER.debug(stock.toString()));
    }

    private static void testTop3Volume() {
        LOGGER.info("Fetching top 3 stocks by volume");
        List<Stock> stocks =
stockRepository.findTop3ByOrderByStVolumeDesc();
    }
}

```



```
2025-07-10T18:39:39.944+05:30 INFO 25184 --- [
restartedMain] com.zaxxer.hikari.pool.HikariPool      :
HikariPool-1 - Added connection conn0: url=jdbc:h2:mem:testdb
user=SA
2025-07-10T18:39:39.948+05:30 INFO 25184 --- [
restartedMain] com.zaxxer.hikari.HikariDataSource      :
HikariPool-1 - Start completed.
2025-07-10T18:39:40.205+05:30 INFO 25184 --- [
restartedMain] o.hibernate.jpa.internal.util.LogHelper      :
HHH000204: Processing PersistenceUnitInfo [name: default]
2025-07-10T18:39:40.321+05:30 INFO 25184 --- [
restartedMain] org.hibernate.Version      :
HHH000412: Hibernate ORM core version 6.6.18.Final
2025-07-10T18:39:40.378+05:30 INFO 25184 --- [
restartedMain] o.h.c.internal.RegionFactoryInitiator      :
HHH000026: Second-level cache disabled
2025-07-10T18:39:40.889+05:30 INFO 25184 --- [
restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo      : No
LoadTimeWeaver setup: ignoring JPA class transformer
2025-07-10T18:39:41.026+05:30 INFO 25184 --- [
restartedMain] org.hibernate.orm.connections.pooling      :
HHH10001005: Database info:
    Database JDBC URL [Connecting through datasource
'HikariDataSource (HikariPool-1)']
    Database driver: undefined/unknown
    Database version: 2.3.232
    Autocommit mode: undefined/unknown
    Isolation level: undefined/unknown
    Minimum pool size: undefined/unknown
    Maximum pool size: undefined/unknown
2025-07-10T18:39:42.472+05:30 INFO 25184 --- [
restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator      :
HHH000489: No JTA platform available (set
'hibernate.transaction.jta.platform' to enable JTA platform
integration)
Hibernate: create table country (co_code varchar(255) not
null, co_name varchar(255), primary key (co_code))
Hibernate: create table department (id integer generated by
default as identity, name varchar(255), primary key (id))
Hibernate: create table employee (id integer generated by
default as identity, date_of_birth timestamp(6), name
varchar(255), permanent boolean, salary float(53),
department_id integer, primary key (id))
Hibernate: alter table if exists stock alter column st_close
set data type float(53)
Hibernate: alter table if exists stock alter column st_code
set data type varchar(255)
Hibernate: alter table if exists stock alter column st_open
set data type float(53)
```



```
Hibernate: alter table if exists stock alter column st_volume
set data type bigint
Hibernate: alter table if exists employee add constraint
FKbejtwvg9bxus2mffsm3swj3u9 foreign key (department_id)
references department
2025-07-10T18:39:42.586+05:30 INFO 25184 --- [
restartedMain] j.LocalContainerEntityManagerFactoryBean :
Initialized JPA EntityManagerFactory for persistence unit
'default'
2025-07-10T18:39:43.773+05:30 INFO 25184 --- [
restartedMain] o.s.b.d.a.OptionalLiveReloadServer :
LiveReload server is running on port 35729
2025-07-10T18:39:43.826+05:30 INFO 25184 --- [
restartedMain] c.c.orm_learn.OrmLearnApplication :
Started OrmLearnApplication in 6.458 seconds (process running
for 7.215)
2025-07-10T18:39:43.836+05:30 INFO 25184 --- [
restartedMain] c.c.orm_learn.OrmLearnApplication :
Fetching Facebook stocks in Sep 2019
Hibernate: select
s1_0.st_id,s1_0.st_close,s1_0.st_code,s1_0.st_date,s1_0.st_ope
n,s1_0.st_volume from stock s1_0 where s1_0.st_code=? and
s1_0.st_date between ? and ?
2025-07-10T18:39:44.147+05:30 INFO 25184 --- [
restartedMain] c.c.orm_learn.OrmLearnApplication :
Fetching Google stocks > 1250
Hibernate: select
s1_0.st_id,s1_0.st_close,s1_0.st_code,s1_0.st_date,s1_0.st_ope
n,s1_0.st_volume from stock s1_0 where s1_0.st_code=? and
s1_0.st_close>?
2025-07-10T18:39:44.153+05:30 INFO 25184 --- [
restartedMain] c.c.orm_learn.OrmLearnApplication :
Fetching top 3 stocks by volume
Hibernate: select
s1_0.st_id,s1_0.st_close,s1_0.st_code,s1_0.st_date,s1_0.st_ope
n,s1_0.st_volume from stock s1_0 order by s1_0.st_volume desc
fetch first ? rows only
2025-07-10T18:39:44.164+05:30 INFO 25184 --- [
restartedMain] c.c.orm_learn.OrmLearnApplication :
Fetching 3 lowest Netflix stocks
Hibernate: select
s1_0.st_id,s1_0.st_close,s1_0.st_code,s1_0.st_date,s1_0.st_ope
n,s1_0.st_volume from stock s1_0 where s1_0.st_code=? order by
s1_0.st_close fetch first ? rows only
2025-07-10T18:39:44.179+05:30 INFO 25184 ---
[ionShutdownHook] j.LocalContainerEntityManagerFactoryBean :
Closing JPA EntityManagerFactory for persistence unit
'default'
```

```
2025-07-10T18:39:44.188+05:30 WARN 25184 ---
[ionShutdownHook] o.s.b.f.support.DisposableBeanAdapter      :
Invocation of destroy method failed on bean with name
'inMemoryDatabaseShutdownExecutor':
org.h2.jdbc.JdbcSQLNonTransientConnectionException: Database
is already closed (to disable automatic closing at VM
shutdown, add ";DB_CLOSE_ON_EXIT=FALSE" to the db URL) [90121-
232]
2025-07-10T18:39:44.189+05:30 INFO 25184 ---
[ionShutdownHook] com.zaxxer.hikari.HikariDataSource          :
HikariPool-1 - Shutdown initiated...
2025-07-10T18:39:44.193+05:30 INFO 25184 ---
[ionShutdownHook] com.zaxxer.hikari.HikariDataSource          :
HikariPool-1 - Shutdown completed.
```

Create payroll tables and bean mapping

```
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

spring.h2.console.enabled=true
spring.h2.console.path=/h2-console
```

```
package com.cognizant.orm_learn;

import jakarta.persistence.*;
import java.util.Date;
import java.util.List;

@Entity
@Table(name = "employee")
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

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

    @Column(name = "salary")
    private double salary;

    @Column(name = "permanent")
    private boolean permanent;

    @Column(name = "date_of_birth")
    private Date dateOfBirth;

    @ManyToOne
    @JoinColumn(name = "department_id")
    private Department department;

    @ManyToMany(fetch = FetchType.EAGER)
```

```

    @JoinTable(name = "employee_skill",
               joinColumns = @JoinColumn(name =
"employee_id"),
               inverseJoinColumns = @JoinColumn(name =
"skill_id"))
    private List<Skill> skills;

    @Override
    public String toString() {
        return "Employee [id=" + id + ", name=" + name + ",
salary=" + salary + ", permanent=" + permanent
        + ", dateOfBirth=" + dateOfBirth + ",
department=" + department + "]";
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    public boolean isPermanent() {
        return permanent;
    }

    public void setPermanent(boolean permanent) {
        this.permanent = permanent;
    }

    public Date getDateOfBirth() {

```

```

        return dateOfBirth;
    }

    public void setDateOfBirth(Date dateOfBirth) {
        this.dateOfBirth = dateOfBirth;
    }

    public Department getDepartment() {
        return department;
    }

    public void setDepartment(Department department) {
        this.department = department;
    }

    public List<Skill> getSkills() {
        return skills;
    }

    public void setSkills(List<Skill> skills) {
        this.skills = skills;
    }

    // Getters, Setters, toString
}

```

```

package com.cognizant.orm_learn;

import jakarta.persistence.*;
import java.util.List;

@Entity
@Table(name = "department")
public class Department {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

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

    @OneToMany(mappedBy = "department")
    private List<Employee> employees;

    @Override
    public String toString() {

```

```

        return "Department [id=" + id + ", name=" + name +
", employees=" + employees + "]";
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public List<Employee> getEmployees() {
        return employees;
    }

    public void setEmployees(List<Employee> employees) {
        this.employees = employees;
    }
}

```

```

package com.cognizant.orm_learn;

import jakarta.persistence.*;
import java.util.List;

@Entity
@Table(name = "skill")
public class Skill {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

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

    @ManyToMany(mappedBy = "skills")
    private List<Employee> employees;
}

```

```

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public List<Employee> getEmployees() {
        return employees;
    }

    public void setEmployees(List<Employee> employees) {
        this.employees = employees;
    }

    @Override
    public String toString() {
        return "Skill [id=" + id + ", name=" + name + ",
employees=" + employees + "]";
    }

    // Getters, Setters, toString
}

```

```

package com.cognizant.orm_learn;

import org.springframework.data.jpa.repository.JpaRepository;
import com.cognizant.orm_learn.Department;

public interface DepartmentRepository extends
JpaRepository<Department, Integer> {}

```

```

package com.cognizant.orm_learn;

import org.springframework.data.jpa.repository.JpaRepository;
import com.cognizant.orm_learn.Skill;

```

```
public interface SkillRepository extends JpaRepository<Skill, Integer> {}
```

```
package com.cognizant.orm_learn;

import org.springframework.data.jpa.repository.JpaRepository;
import com.cognizant.orm_learn.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {}
```

```
package com.cognizant.orm_learn;

import com.cognizant.orm_learn.Department;
import com.cognizant.orm_learn.Employee;
import com.cognizant.orm_learn.Skill;
import com.cognizant.orm_learn.EmployeeRepository;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.util.List;

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner
{

    @Autowired
    private EmployeeRepository employeeRepository;

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

    @Override
    public void run(String... args) throws Exception {
        System.out.println("Getting employee details with department and skills...");

        Employee employee = employeeRepository.findById(1).orElse(null);

        if (employee != null) {
```



```
2025-07-10T08:32:50.353+05:30 INFO 33808 --- [
restartedMain] com.zaxxer.hikari.HikariDataSource      :
HikariPool-1 - Starting...
2025-07-10T08:32:50.726+05:30 INFO 33808 --- [
restartedMain] com.zaxxer.hikari.pool.HikariPool      :
HikariPool-1 - Added connection conn0: url=jdbc:h2:mem:testdb
user=SA
2025-07-10T08:32:50.730+05:30 INFO 33808 --- [
restartedMain] com.zaxxer.hikari.HikariDataSource      :
HikariPool-1 - Start completed.
2025-07-10T08:32:50.975+05:30 INFO 33808 --- [
restartedMain] o.hibernate.jpa.internal.util.LogHelper      :
HHH000204: Processing PersistenceUnitInfo [name: default]
2025-07-10T08:32:51.044+05:30 INFO 33808 --- [
restartedMain] org.hibernate.Version                          :
HHH000412: Hibernate ORM core version 6.6.18.Final
2025-07-10T08:32:51.086+05:30 INFO 33808 --- [
restartedMain] o.h.c.internal.RegionFactoryInitiator      :
HHH000026: Second-level cache disabled
2025-07-10T08:32:51.465+05:30 INFO 33808 --- [
restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo      : No
LoadTimeWeaver setup: ignoring JPA class transformer
2025-07-10T08:32:51.582+05:30 INFO 33808 --- [
restartedMain] org.hibernate.orm.connections.pooling      :
HHH10001005: Database info:
    Database JDBC URL [Connecting through datasource
'HikariDataSource (HikariPool-1)']
    Database driver: undefined/unknown
    Database version: 2.3.232
    Autocommit mode: undefined/unknown
    Isolation level: undefined/unknown
    Minimum pool size: undefined/unknown
    Maximum pool size: undefined/unknown
2025-07-10T08:32:52.863+05:30 INFO 33808 --- [
restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator      :
HHH000489: No JTA platform available (set
'hibernate.transaction.jta.platform' to enable JTA platform
integration)
Hibernate: create table country (co_code varchar(255) not
null, co_name varchar(255), primary key (co_code))
Hibernate: alter table if exists employee alter column
date_of_birth set data type timestamp(6)
2025-07-10T08:32:52.944+05:30 INFO 33808 --- [
restartedMain] j.LocalContainerEntityManagerFactoryBean :
Initialized JPA EntityManagerFactory for persistence unit
'default'
2025-07-10T08:32:54.018+05:30 INFO 33808 --- [
restartedMain] o.s.b.d.a.OptionalLiveReloadServer      :
LiveReload server is running on port 35729
```

```

2025-07-10T08:32:54.051+05:30 INFO 33808 --- [
restartedMain] c.c.orm_learn.OrmLearnApplication      :
Started OrmLearnApplication in 5.91 seconds (process running
for 6.738)
Getting employee details with department and skills...
Hibernate: select
e1_0.id,e1_0.date_of_birth,d1_0.id,d1_0.name,e1_0.name,e1_0.pe
rmanent,e1_0.salary,s1_0.employee_id,s1_1.id,s1_1.name from
employee e1_0 left join department d1_0 on
d1_0.id=e1_0.department_id left join employee_skill s1_0 on
e1_0.id=s1_0.employee_id left join skill s1_1 on
s1_1.id=s1_0.skill_id where e1_0.id=?
Name: Ram
Salary: 50000.0
Department: Engineering
Skills:
    - Java
    - Python
2025-07-10T08:32:54.259+05:30 INFO 33808 ---
[ionShutdownHook] j.LocalContainerEntityManagerFactoryBean :
Closing JPA EntityManagerFactory for persistence unit
'default'
2025-07-10T08:32:54.268+05:30 WARN 33808 ---
[ionShutdownHook] o.s.b.f.support.DisposableBeanAdapter      :
Invocation of destroy method failed on bean with name
'inMemoryDatabaseShutdownExecutor':
org.h2.jdbc.JdbcSQLNonTransientConnectionException: Database
is already closed (to disable automatic closing at VM
shutdown, add ";DB_CLOSE_ON_EXIT=FALSE" to the db URL) [90121-
232]
2025-07-10T08:32:54.269+05:30 INFO 33808 ---
[ionShutdownHook] com.zaxxer.hikari.HikariDataSource          :
HikariPool-1 - Shutdown initiated...
2025-07-10T08:32:54.274+05:30 INFO 33808 ---
[ionShutdownHook] com.zaxxer.hikari.HikariDataSource          :
HikariPool-1 - Shutdown completed.

```

Demonstrate writing Hibernate Query Language and Native Query

```

DROP TABLE IF EXISTS employee_skill;
DROP TABLE IF EXISTS employee;
DROP TABLE IF EXISTS skill;
DROP TABLE IF EXISTS department;

CREATE TABLE department (
    id INT PRIMARY KEY,

```

```

        name VARCHAR(255)
    );

CREATE TABLE skill (
    id INT PRIMARY KEY,
    name VARCHAR(255)
);

CREATE TABLE employee (
    id INT PRIMARY KEY,
    name VARCHAR(255),
    salary DECIMAL(10, 2),
    permanent BOOLEAN,
    date_of_birth DATE,
    department_id INT,
    FOREIGN KEY (department_id) REFERENCES department(id)
);

CREATE TABLE employee_skill (
    employee_id INT NOT NULL,
    skill_id INT NOT NULL,
    PRIMARY KEY (employee_id, skill_id),
    CONSTRAINT fk_emp FOREIGN KEY (employee_id) REFERENCES
employee(id),
    CONSTRAINT fk_skill FOREIGN KEY (skill_id) REFERENCES
skill(id)
);

```

```

INSERT INTO department (id, name) VALUES (1, 'Technology'),
(2, 'HR');

```

```

INSERT INTO skill (id, name) VALUES (1, 'Java'), (2, 'SQL'),
(3, 'Spring Boot');

```

```

INSERT INTO employee (id, name, salary, permanent,
date_of_birth, department_id)
VALUES (1, 'Alice', 60000, true, '1995-02-15', 1),
(2, 'Bob', 50000, false, '1996-07-23', 1),
(3, 'Carol', 70000, true, '1990-03-10', 2);

```

```

INSERT INTO employee_skill (employee_id, skill_id) VALUES
(1, 1), (1, 2), (2, 2), (3, 1), (3, 3);

```

```

package com.cognizant.orm_learn;

```

```

import jakarta.persistence.*;
import java.util.Date;

```

```

import java.util.List;

@Entity
@Table(name = "employee")
public class Employee {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String name;
    private double salary;
    private boolean permanent;
    @Temporal(TemporalType.DATE)
    private Date dateOfBirth;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "department_id")
    private Department department;

    @ManyToMany(fetch = FetchType.LAZY)
    @JoinTable(name = "employee_skill",
        joinColumns = @JoinColumn(name = "employee_id"),
        inverseJoinColumns = @JoinColumn(name = "skill_id"))
    private List<Skill> skillList;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    public boolean isPermanent() {
        return permanent;
    }
}

```

```

    }

    public void setPermanent(boolean permanent) {
        this.permanent = permanent;
    }

    public Date getDateOfBirth() {
        return dateOfBirth;
    }

    public void setDateOfBirth(Date dateOfBirth) {
        this.dateOfBirth = dateOfBirth;
    }

    public Department getDepartment() {
        return department;
    }

    public void setDepartment(Department department) {
        this.department = department;
    }

    public List<Skill> getSkillList() {
        return skillList;
    }

    public void setSkillList(List<Skill> skillList) {
        this.skillList = skillList;
    }

    @Override
    public String toString() {
        return "Employee [id=" + id + ", name=" + name + ",
salary=" + salary + ", permanent=" + permanent
        + ", dateOfBirth=" + dateOfBirth + ",
department=" + department + ", skillList=" + skillList + "]";
    }
}

```

```

package com.cognizant.orm_learn;
import jakarta.persistence.*;

@Entity @Table(name = "skill")
public class Skill {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String name;
}

```

```

    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    @Override
    public String toString() {
        return "Skill [id=" + id + ", name=" + name + "]";
    }
}

```

```

package com.cognizant.orm_learn;

import jakarta.persistence.*;

@Entity @Table(name = "department")
public class Department {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String name;
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    @Override
    public String toString() {
        return "Department [id=" + id + ", name=" + name +
    "]"
    }
}

```

```
package com.cognizant.orm_learn;

import com.cognizant.orm_learn.Department;
import org.springframework.data.jpa.repository.JpaRepository;

public interface DepartmentRepository extends
JpaRepository<Department, Integer> { }
```

```
package com.cognizant.orm_learn;

import org.springframework.data.jpa.repository.JpaRepository;

public interface SkillRepository extends JpaRepository<Skill,
Integer> { }
```

```
package com.cognizant.orm_learn;

import com.cognizant.orm_learn.Employee;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import java.util.List;

public interface EmployeeRepository extends
JpaRepository<Employee, Integer> {
    // HQL / JPQL
    @Query("SELECT e FROM Employee e WHERE e.permanent =
true")
    List<Employee> findPermanent();

    // HQL with JOIN FETCH to avoid N+1
    @Query("SELECT DISTINCT e FROM Employee e " +
        "JOIN FETCH e.department JOIN FETCH e.skillList " +
        "WHERE e.permanent = true")
    List<Employee> findPermanentWithDetails();

    // HQL aggregate example: average salary per department
    @Query("SELECT e.department.name, AVG(e.salary) FROM
Employee e GROUP BY e.department.name")
    List<Object[]> avgSalaryByDepartment();

    // Native query: return all permanent employees
    @Query(value = "SELECT * FROM employee WHERE permanent =
true", nativeQuery = true)
    List<Employee> findPermanentNative();
}
```

```
package com.cognizant.orm_learn;
```



```

import com.cognizant.orm_learn.Employee;
import com.cognizant.orm_learn.EmployeeRepository;
import org.springframework.stereotype.Service;
import jakarta.transaction.Transactional;
import java.util.List;

@Service
public class EmployeeService {
    private final EmployeeRepository repo;
    public EmployeeService(EmployeeRepository repo) {
        this.repo = repo;
    }

    @Transactional public List<Employee> getPermanent() {
        return repo.findPermanent();
    }

    @Transactional public List<Employee>
    getPermanentWithDetails() {
        return repo.findPermanentWithDetails();
    }

    @Transactional public List<Object[]> getAvgSalaryByDept()
    {
        return repo.avgSalaryByDepartment();
    }

    @Transactional public List<Employee> getPermanentNative()
    {
        return repo.findPermanentNative();
    }
}

```

```

package com.cognizant.orm_learn;

import com.cognizant.orm_learn.Department;
import com.cognizant.orm_learn.DepartmentRepository;
import jakarta.transaction.Transactional;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Service;
import org.springframework.beans.factory.annotation.Autowired;

@Service
public class DepartmentService {
    private static final Logger LOGGER =
    LoggerFactory.getLogger(DepartmentService.class);
}

```

```

    @Autowired
    private DepartmentRepository departmentRepository;

    @Transactional
    public Department get(int id) {
        LOGGER.info("Start");
        return departmentRepository.findById(id).get();
    }

    @Transactional
    public void save(Department department) {
        LOGGER.info("Start");
        departmentRepository.save(department);
        LOGGER.info("End");
    }
}

```

```

package com.cognizant.orm_learn;

import com.cognizant.orm_learn.Employee;
import com.cognizant.orm_learn.EmployeeService;
import org.slf4j.*;
import org.springframework.boot.*;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import java.util.List;

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner
{
    private static final Logger LOGGER =
LoggerFactory.getLogger(OrmLearnApplication.class);
    private final EmployeeService employeeService;

    public OrmLearnApplication(EmployeeService
employeeService) {
        this.employeeService = employeeService;
    }

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

    @Override
    public void run(String... args) {

```

```

        LOGGER.info("=== HQL without fetch ===");
        employeeService.getPermanent().forEach(e ->
LOGGER.debug("Emp: {}", e));

        LOGGER.info("=== HQL with JOIN FETCH ===");
        employeeService.getPermanentWithDetails()
            .forEach(e -> LOGGER.debug("Emp: {} Dept: {}
Skills: {}", e.getName(), e.getDepartment(),
e.getSkillList()));

        LOGGER.info("=== HQL Aggregate (Avg salary per dept)
===");
        employeeService.getAvgSalaryByDept()
            .forEach(row -> LOGGER.debug("Dept: {} Avg Salary:
{}", row[0], row[1]));

        LOGGER.info("=== Native SQL Query ===");
        employeeService.getPermanentNative().forEach(e ->
LOGGER.debug("Emp (Native): {}", e));
    }
}

```

OUTPUT:

```

  ____
 /  __ \
(  )  /
 \  __/
  ____

:: Spring Boot ::                (v3.5.3)

2025-07-10T19:54:26.803+05:30 INFO 23448 --- [
restartedMain] c.c.orm_learn.OrmLearnApplication      :
Starting OrmLearnApplication using Java 21.0.7 with PID 23448
(C:\Users\admin\Downloads\orm-learn\orm-learn\target\classes
started by admin in C:\Users\admin\Downloads\orm-learn\orm-
learn)
2025-07-10T19:54:26.806+05:30 INFO 23448 --- [
restartedMain] c.c.orm_learn.OrmLearnApplication      : No
active profile set, falling back to 1 default profile:
"default"
2025-07-10T19:54:26.886+05:30 INFO 23448 --- [
restartedMain] .e.DevToolsPropertyDefaultsPostProcessor :
Devtools property defaults active! Set 'spring.devtools.add-
properties' to 'false' to disable

```

```
2025-07-10T19:54:27.877+05:30 INFO 23448 --- [
restartedMain] .s.d.r.c.RepositoryConfigurationDelegate :
Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2025-07-10T19:54:27.986+05:30 INFO 23448 --- [
restartedMain] .s.d.r.c.RepositoryConfigurationDelegate :
Finished Spring Data repository scanning in 90 ms. Found 5 JPA
repository interfaces.
2025-07-10T19:54:28.570+05:30 INFO 23448 --- [
restartedMain] com.zaxxer.hikari.HikariDataSource      :
HikariPool-1 - Starting...
2025-07-10T19:54:28.960+05:30 INFO 23448 --- [
restartedMain] com.zaxxer.hikari.pool.HikariPool      :
HikariPool-1 - Added connection conn0: url=jdbc:h2:mem:testdb
user=SA
2025-07-10T19:54:28.963+05:30 INFO 23448 --- [
restartedMain] com.zaxxer.hikari.HikariDataSource      :
HikariPool-1 - Start completed.
2025-07-10T19:54:29.267+05:30 INFO 23448 --- [
restartedMain] o.hibernate.jpa.internal.util.LogHelper  :
HHH000204: Processing PersistenceUnitInfo [name: default]
2025-07-10T19:54:29.355+05:30 INFO 23448 --- [
restartedMain] org.hibernate.Version                      :
HHH000412: Hibernate ORM core version 6.6.18.Final
2025-07-10T19:54:29.417+05:30 INFO 23448 --- [
restartedMain] o.h.c.internal.RegionFactoryInitiator    :
HHH000026: Second-level cache disabled
2025-07-10T19:54:29.928+05:30 INFO 23448 --- [
restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo    : No
LoadTimeWeaver setup: ignoring JPA class transformer
2025-07-10T19:54:30.078+05:30 INFO 23448 --- [
restartedMain] org.hibernate.orm.connections.pooling   :
HHH10001005: Database info:
    Database JDBC URL [Connecting through datasource
'HikariDataSource (HikariPool-1)']
    Database driver: undefined/unknown
    Database version: 2.3.232
    Autocommit mode: undefined/unknown
    Isolation level: undefined/unknown
    Minimum pool size: undefined/unknown
    Maximum pool size: undefined/unknown
2025-07-10T19:54:31.892+05:30 INFO 23448 --- [
restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator     :
HHH000489: No JTA platform available (set
'hibernate.transaction.jta.platform' to enable JTA platform
integration)
Hibernate: create table country (co_code varchar(255) not
null, co_name varchar(255), primary key (co_code))
Hibernate: alter table if exists employee alter column salary
set data type float(53)
```

```

Hibernate: create table stock (st_id integer generated by
default as identity, st_close float(53), st_code varchar(255),
st_date date, st_open float(53), st_volume bigint, primary key
(st_id))
2025-07-10T19:54:32.008+05:30 INFO 23448 --- [
restartedMain] j.LocalContainerEntityManagerFactoryBean :
Initialized JPA EntityManagerFactory for persistence unit
'default'
2025-07-10T19:54:32.670+05:30 INFO 23448 --- [
restartedMain] o.s.d.j.r.query.QueryEnhancerFactory :
Hibernate is in classpath; If applicable, HQL parser will be
used.
2025-07-10T19:54:34.262+05:30 INFO 23448 --- [
restartedMain] o.s.b.d.a.OptionalLiveReloadServer :
LiveReload server is running on port 35729
2025-07-10T19:54:34.290+05:30 INFO 23448 --- [
restartedMain] c.c.orm_learn.OrmLearnApplication :
Started OrmLearnApplication in 8.133 seconds (process running
for 8.866)
2025-07-10T19:54:34.302+05:30 INFO 23448 --- [
restartedMain] c.c.orm_learn.OrmLearnApplication : ===
HQL without fetch ===
Hibernate: select
e1_0.id,e1_0.date_of_birth,e1_0.department_id,e1_0.name,e1_0.p
ermanent,e1_0.salary from employee e1_0 where
e1_0.permanent=true
2025-07-10T19:54:34.566+05:30 INFO 23448 --- [
restartedMain] c.c.orm_learn.OrmLearnApplication : ===
HQL with JOIN FETCH ===
Hibernate: select distinct
e1_0.id,e1_0.date_of_birth,d1_0.id,d1_0.name,e1_0.name,e1_0.pe
rmanent,e1_0.salary,s11_0.employee_id,s11_1.id,s11_1.name from
employee e1_0 join department d1_0 on
d1_0.id=e1_0.department_id join employee_skill s11_0 on
e1_0.id=s11_0.employee_id join skill s11_1 on
s11_1.id=s11_0.skill_id where e1_0.permanent=true
2025-07-10T19:54:34.605+05:30 INFO 23448 --- [
restartedMain] c.c.orm_learn.OrmLearnApplication : ===
HQL Aggregate (Avg salary per dept) ===
Hibernate: select d1_0.name,avg(e1_0.salary) from employee
e1_0 join department d1_0 on d1_0.id=e1_0.department_id group
by d1_0.name
2025-07-10T19:54:34.629+05:30 INFO 23448 --- [
restartedMain] c.c.orm_learn.OrmLearnApplication : ===
Native SQL Query ===
Hibernate: SELECT * FROM employee WHERE permanent = true
2025-07-10T19:54:34.688+05:30 INFO 23448 ---
[ionShutdownHook] j.LocalContainerEntityManagerFactoryBean :

```

```
Closing JPA EntityManagerFactory for persistence unit
'default'
2025-07-10T19:54:34.695+05:30  WARN 23448 ---
[ionShutdownHook] o.s.b.f.support.DisposableBeanAdapter      :
Invocation of destroy method failed on bean with name
'inMemoryDatabaseShutdownExecutor':
org.h2.jdbc.JdbcSQLNonTransientConnectionException: Database
is already closed (to disable automatic closing at VM
shutdown, add ";DB_CLOSE_ON_EXIT=FALSE" to the db URL) [90121-
232]
2025-07-10T19:54:34.696+05:30  INFO 23448 ---
[ionShutdownHook] com.zaxxer.hikari.HikariDataSource          :
HikariPool-1 - Shutdown initiated...
2025-07-10T19:54:34.699+05:30  INFO 23448 ---
[ionShutdownHook] com.zaxxer.hikari.HikariDataSource          :
HikariPool-1 - Shutdown completed.
```