spring.application.name=SpringBoot4-JDBC  
spring.h2.console.enabled=true  
spring.datasource.url=jdbc:h2:mem:testdb

-------------

package com.trg.model;  
  
public class Account {  
 int acctId;  
 String aname, type;  
 double bal;  
  
 *//get/set, cons, toString()* public Account(int acctId, String aname, String type, double bal) {  
 this.acctId = acctId;  
 this.aname = aname;  
 this.type = type;  
 this.bal = bal;  
 }  
 public int getAcctId() {  
 return acctId;  
 }  
  
 public double getBal() {  
 return bal;  
 }  
  
 public String getAname() {  
 return aname;  
 }  
  
 public String getType() {  
 return type;  
 }  
  
 @Override  
 public String toString() {  
 return "Account{" +  
 "acctId=" + acctId +  
 ", aname='" + aname + '\'' +  
 ", type='" + type + '\'' +  
 ", bal=" + bal +  
 '}';  
 }  
}

--------------------------------------------

package com.trg.dao;  
  
import com.trg.model.Account;  
  
import java.util.List;  
  
public interface AccountDaoIntf {  
 public int count();  
 public int save(Account account);  
 public int update(Account account);  
 public int deleteById(int id);  
 public List<Account> findAll();  
 public List<Account> findByName(String name);  
 public String getNameById(int i);  
  
}

---------------------------------------------

package com.trg.dao;  
  
import com.trg.model.Account;  
import jakarta.annotation.PostConstruct;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.jdbc.core.JdbcTemplate;  
import org.springframework.stereotype.Repository;  
  
import java.util.List;  
  
@Repository  
public class AccountDaoImpl implements AccountDaoIntf {  
  
 @Autowired  
 JdbcTemplate jdbcTemplate;  
  
 @PostConstruct  
 public void init(){  
 System.*out*.println("jdbcTemplate : " + jdbcTemplate);  
 String sql1 = "DROP TABLE accounts IF EXISTS";  
 String sql2 = "CREATE TABLE accounts(acctid number(3), aname VARCHAR(20), type varchar(15), bal number(7, 2))";  
 jdbcTemplate.execute(sql1);  
 jdbcTemplate.execute(sql2);  
 }  
  
  
 public int save(Account ac) {  
 String sql = "insert into accounts values(?,?,?,?)";  
 return jdbcTemplate.update(sql, ac.getAcctId(), ac.getAname(),  
 ac.getType(), ac.getBal());  
 }  
  
 public int update(Account ac) {  
 String sql = "update accounts set bal = ? where acctid = ?";  
 return jdbcTemplate.update(sql, ac.getBal(), ac.getAcctId());  
 }  
  
 public int deleteById(int id) {  
 return jdbcTemplate.update("delete from accounts where acctid = ?", id);  
 }  
  
 public List<Account> findAll() {  
 return jdbcTemplate.query("select \* from accounts", (rs, rowno) ->  
 new Account(rs.getInt("acctid"), rs.getString("aname"),  
 rs.getString("type"), rs.getDouble("bal")));  
 }  
 public int count() {  
 String sql = "select count(\*) from accounts";  
 return jdbcTemplate.queryForObject(sql, Integer.class);  
 }  
  
 @Override  
 public String getNameById(int id) {  
 String sql = "select \* from accounts where acctid=?";  
 Account ac = jdbcTemplate.queryForObject(sql , new Object[] {id},  
 (rs, rowNum) -> new Account(rs.getInt("acctid"),  
 rs.getString("aname"), rs.getString("type"), rs.getDouble("bal")));  
 return ac.getAname();  
 }  
  
 @Override  
 public List<Account> findByName(String name) {  
 String sql = "select \* from accounts where aname=?";  
 List<Account> list = jdbcTemplate.query(sql , new Object[] {name},  
 (rs, rowNum) -> new Account(rs.getInt("acctid"),  
 rs.getString("aname"), rs.getString("type"), rs.getDouble("bal")));  
 return list;  
 }  
  
  
}

----------------------------------------------

package com.trg.client;  
  
import com.trg.dao.AccountDaoImpl;  
import com.trg.dao.AccountDaoIntf;  
import com.trg.model.Account;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Component;  
  
import java.util.Arrays;  
import java.util.List;  
  
@Component  
public class UseAccount {  
  
 @Autowired  
 AccountDaoImpl dao;  
  
 public void accountOps(){  
 *//dao.init();* List<Account> accounts = Arrays.*asList*(  
 new Account(1, "Account-1", "savings", 50000),  
 new Account(2, "Account-2", "savings", 20000),  
 new Account(3, "Account-3", "current", 35000),  
 new Account(4, "Account-4", "current", 20000));  
 System.*out*.println("------- saving all records -------------");  
 accounts.forEach(account -> { dao.save(account); });  
 System.*out*.println("[COUNT] Total accounts: " + dao.count());  
 System.*out*.println("[LIST ALL] : " + dao.findAll());  
 System.*out*.println("[LIST BY NAME] : " +dao.findByName("Account-4"));  
 System.*out*.println("[FIND BY ID] : " +dao.getNameById(3));  
 Account newacct = new Account(5, "Account-5", "savings", 60000);  
 dao.save(newacct);  
 System.*out*.println("[LIST ALL] : " + dao.findAll());  
 }  
}

-----------------------------------------

package com.trg;  
  
import com.trg.client.UseAccount;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.ApplicationContext;  
  
@SpringBootApplication  
public class SpringBoot4JdbcApplication {  
  
 public static void main(String[] args) {  
 ApplicationContext ctx = SpringApplication.*run*(SpringBoot4JdbcApplication.class, args);  
 UseAccount test = ctx.getBean("useAccount", UseAccount.class);  
 test.accountOps();  
  
 }  
  
}

---------------------------------------------------

--------------- JpaRepository -------------------------

*/\* data.sql \*/  
insert into courses values(1,'Spring Microservices','Shrilata', 5)  
insert into courses values(2,'AWS basics','Veena', 3)  
insert into courses values(3,'Learning Javascript','Shrilata', 3)  
insert into courses values(4,'Advanced Javascript','Mahima', 3)  
insert into courses values(5,'MERN','Veena', 15)*

-----------------------------

*/\* schema.sql \*/*drop table if exists courses;  
  
create table courses (  
 id INT PRIMARY KEY AUTO\_INCREMENT,  
 cname varchar(25),  
 trainer varchar(15),  
 duration int  
);

@ControllerAdvice  
public class GlobalExceptionHandler {  
  
 @ExceptionHandler(MethodArgumentNotValidException.class)  
 public ResponseEntity<?> notValid(MethodArgumentNotValidException ex, HttpServletRequest request) {  
 List<String> errors = new ArrayList<>();  
  
 ex.getAllErrors().forEach(err -> errors.add(err.getDefaultMessage()));  
  
 Map<String, List<String>> result = new HashMap<>();  
 result.put("errors", errors);  
  
 return new ResponseEntity<>(result, HttpStatus.*BAD\_REQUEST*);  
 }  
}