BEANCONFIG FILE

```
package com.psl.training.config;

import java.util.Properties;

import javax.sql.DataSource;

import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.datasource.DriverManagerDataSource;
import org.springframework.orm.hibernate5.HibernateTransactionManager;
import org.springframework.orm.hibernate5.LocalSessionFactoryBean;
import org.springframework.transaction.TransactionManager;

@Configuration // this annotation will tell spring that this class provides configurations
@ComponentScan(basePackages = "com.psl.training")
//@PropertySource(value = "classpath:/application.properties")
public class BeanConfig {

	// Establish Connection

	// resposible for making Connection with database

	/*
```

```
	 * @Value("${mysql.driver.classname}") String driverClassName;
	 *
	 * @Value("${mysql.url}") String url;
	 *
	 * @Value("${mysql.username}") String username;
	 *
	 * @Value("${mysql.password}") String password;
	 */

	@Bean
	public DataSource getDataSource() {
		DriverManagerDataSource datasource = new DriverManagerDataSource();
		datasource.setDriverClassName("com.mysql.cj.jdbc.Driver");

	datasource.setUrl("jdbc:mysql://localhost:3306/empsystem?allowPublicKeyRetrieval=true&use
SSL=false");
		datasource.setUsername("root");
		datasource.setPassword("root");
		return datasource;
	}

	@Bean
	public LocalSessionFactoryBean getSessionFactoryBean() {

		LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();
		sessionFactory.setDataSource(getDataSource());

		// set the scan path where JPA Entity annotations are present
		sessionFactory.setPackagesToScan("com.psl.training.model");
```

```
		// setting hibernate specific properties like show_sql
		Properties props = new Properties();

		// to print all the sql statements generated by hibernate
		props.put("hibernate.show_sql", "true");

		// Create Tables at back end with the help of Entity annotations present
		props.put("hibernate.hbm2ddl.auto", "update"); // it will create table if not present and alter table if fields
	// are not present as mentioned in Entity

		sessionFactory.setHibernateProperties(props);
		return sessionFactory;

	}

	@Bean
	public TransactionManager getTransactionManager() {
		SessionFactory sessionFactory = getSessionFactoryBean().getObject();
		HibernateTransactionManager transactionManager = new HibernateTransactionManager();
		transactionManager.setSessionFactory(sessionFactory);
		return transactionManager;
	}

}
```

REQUIRED DEPENDECIES

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-core</artifactId>
			<version>5.3.25</version>
		</dependency>
		<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-context</artifactId>
			<version>5.3.25</version>
		</dependency>
		<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
		<dependency>
			<groupId>mysql</groupId>
			<artifactId>mysql-connector-java</artifactId>
			<version>8.0.29</version>
		</dependency>
		<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
		<dependency>
			<groupId>org.hibernate</groupId>
			<artifactId>hibernate-core</artifactId>
			<version>5.6.15.Final</version>
		</dependency>
		<!-- https://mvnrepository.com/artifact/org.springframework/spring-orm -->
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-orm</artifactId>
			<version>5.3.25</version>
```

```
		</dependency>
```

EMPLOYEE ENTITY CLASS

```
package com.psl.training.model;

import java.sql.Date;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;


@Entity // declaring this is Java object related with RDBMS table

@Table(name = "emp") /// mapping table with Java Object (Optional if table name is matching with ///

                                                    /// class name

public class Employee {


	// @GeneratedValue(strategy = GenerationType.IDENTITY) // for generating primary

	// key values automatically this is optional


	@Id

	private int empid;

	private String empname;

	private String city;

	private Date joindate;


	public Employee() {

		// TODO Auto-generated constructor stub
```

```
}

public Employee(int empid, String empname, String city, Date joindate) {
        super();
        this.empid = empid;
        this.empname = empname;
        this.city = city;
        this.joindate = joindate;
}

public int getEmpid() {
        return empid;
}

public void setEmpid(int empid) {
        this.empid = empid;
}

public String getEmpname() {
        return empname;
}

public void setEmpname(String empname) {
        this.empname = empname;
}

public String getCity() {
        return city;
}
```

```
	public void setCity(String city) {
		this.city = city;
	}

	public Date getJoindate() {
		return joindate;
	}

	public void setJoindate(Date joindate) {
		this.joindate = joindate;
	}

	@Override
	public String toString() {
		return "Employee [empid=" + empid + ", empname=" + empname + ", city=" + city + ",
joindate=" + joindate + "]";
	}

}
```

EMPLOYEEDAO INTERFACE

```
package com.psl.training.dao;

import java.util.List;
import com.psl.training.model.Employee;

public interface EmployeeDAO1 {
	public boolean insertEmployee(Employee emp);// insert
```

```
	public Employee getEmployeeById(int id);// select --gives only one emp information

	public List<Employee> getAllEmployees();// all emp information

	public boolean updateEmployee(Employee emp);// update

	public boolean deleteEmployee(int id);// delete
}
```

EMPLOYEEDAOIMPL CLASS

```
package com.psl.training.dao;

import java.util.List;

import javax.persistence.EntityManager;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import com.psl.training.model.Employee;

// get connected to db
@Repository
public class EmployeeDAOImpl implements EmployeeDAO1 {

	@Autowired
```

```
SessionFactory sessionFactory;

public boolean insertEmployee(Employee emp) {

	Session session = sessionFactory.openSession();
	session.beginTransaction(); // transaction started
	session.save(emp);// insert query
	session.getTransaction().commit();
	return true;

}

public Employee getEmployeeById(int id) {

	// hibernate will use session for performing db operations
	Session session = sessionFactory.openSession();
	return session.get(Employee.class, id);
}

public List<Employee> getAllEmployees() {
	Session session = sessionFactory.openSession();
	return session.createQuery("from Employee database", Employee.class).list();

}

public boolean updateEmployee(Employee emp) {
	try {
		Session session = sessionFactory.openSession();
		session.beginTransaction(); // transaction started
```

```
                session.save(emp); // if record not present it will insert else it will update
                session.getTransaction().commit();
                return true;
            } catch (Exception e) {
                throw e;
            }
        }

        public boolean deleteEmployee(int id) {
            Session session = sessionFactory.openSession();
            Employee e = session.get(Employee.class, id);
            if (e == null)
                throw new RuntimeException("Resource Not found");
            session.delete(e);
            return true;
        }
}
```

EMPLOYEESERVICE INTERFACE

```
package com.psl.training.service;

import java.util.List;

import com.psl.training.model.Employee;

public interface EmployeeService {
        public boolean insertEmployee(Employee emp);
        public Employee getEmployeeById(int id);
        public List<Employee> getAllEmployees();
```

```
	public boolean updateEmployee(Employee emp);
	public boolean deleteEmployee(int id);
}
```

EMPLOYEESERVICEIMPL CLASS

```
package com.psl.training.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.psl.training.dao.EmployeeDAO1;
import com.psl.training.model.Employee;

// interact with dao layer
// may handle transactions
@Service
public class EmployeeServiceImpl implements EmployeeService {

	@Autowired
	EmployeeDAO1 dao;

	public boolean insertEmployee(Employee emp) {
		return dao.insertEmployee(emp);
	}
```

```
	public Employee getEmployeeById(int id) {
		return dao.getEmployeeById(id);
	}

	public List<Employee> getAllEmployees() {
		return dao.getAllEmployees();
	}

	public boolean updateEmployee(Employee emp) {
		return false;

	}

	public boolean deleteEmployee(int empid) {
		return false;

	}

}
```

MAIN CLASS

```
package com.psl.training.test;

import java.sql.Date;
import java.time.LocalDate;

import javax.sql.DataSource;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
```

```
import org.springframework.jdbc.core.JdbcTemplate;
import com.psl.training.config.BeanConfig;
import com.psl.training.model.Employee;
import com.psl.training.service.EmployeeService;
import com.psl.training.service.EmployeeServiceImpl;

public class Main {

	public static void main(String[] args) {

		ApplicationContext context = new
AnnotationConfigApplicationContext(BeanConfig.class);

		Employee emp = new Employee(203, "ABC", "Goa", Date.valueOf(LocalDate.now()));

		EmployeeService service = context.getBean(EmployeeServiceImpl.class);

		System.out.println("Data inserted successfully" + service.insertEmployee(emp));

		// System.out.println(context.getBean(DataSource.class));
		//System.out.println(context.getBean(JdbcTemplate.class));
		// System.out.println(service.getEmployeeById(101));
		// service.getAllEmployees().forEach(System.out::println);
		// System.out.println(service.getAllEmployees());

	}

}
```