1. Spring Dependency Injection

MessageService.java

```java
public class MessageService {
    public String getMessage() {
        return "Hello from MessageService!";
    }
}
```

UserController.java

```java
public class UserController {

    private MessageService messageService;

    // Default constructor
    public UserController() {}

    // Constructor Injection
    public UserController(MessageService messageService) {
        this.messageService = messageService;
    }

    // Setter Injection
    public void setMessageService(MessageService messageService) {
        this.messageService = messageService;
    }

    // Field Injection
    private MessageService fieldMessageService;
    public void setFieldMessageService(MessageService fieldMessageService) {
        this.fieldMessageService = fieldMessageService;
    }

    // Method to print message
    public String printMessage() {
        return messageService.getMessage();
    }

    // Method for field-based message service
    public String printMessageFromField() {
        return fieldMessageService.getMessage();
    }
}
```

Main.java

```java
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main {
    public static void main(String[] args) {
        // Load Spring application context from beans.xml
        ApplicationContext context = new
ClassPathXmlApplicationContext("beans.xml");

        // Retrieve the UserController beans using different injection methods
        UserController userConstructor =
context.getBean("userControllerConstructor", UserController.class);
        UserController userSetter = context.getBean("userControllerSetter",
UserController.class);
        UserController userField = context.getBean("userControllerField",
UserController.class);
```

```
        // Output the messages from each UserController instance
        System.out.println("Constructor Injection: " +
userConstructor.printMessage());
        System.out.println("Setter Injection: " + userSetter.printMessage());
        System.out.println("Field Injection: " +
userField.printMessageFromField());
    }
}
```

beans.xml

```
<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-4.3.xsd">

    <!-- MessageService Bean -->
    <bean id="messageService" class="com.example.di.MessageService" />

    <!-- UserController Bean using Constructor Injection -->
    <bean id="userControllerConstructor" class="com.example.di.UserController">
        <constructor-arg ref="messageService" />
    </bean>

    <!-- UserController Bean using Setter Injection -->
    <bean id="userControllerSetter" class="com.example.di.UserController">
        <property name="messageService" ref="messageService" />
    </bean>

    <!-- UserController Bean using Field Injection -->
    <bean id="userControllerField" class="com.example.di.UserController">
        <property name="fieldMessageService" ref="messageService" />
    </bean>
</beans>
```

2. Spring Bean Lifecycle

DatabaseConnection.java

```
import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import org.springframework.stereotype.Component;

@Component
public class DatabaseConnection {

    // Constructor
    public DatabaseConnection() {
        System.out.println("DatabaseConnection object created!");
    }

    // @PostConstruct annotated method will be called after the bean's
initialization.
    @PostConstruct
    public void init() {
        System.out.println("Database connection initialized");
    }

    // @PreDestroy annotated method will be called before the bean is destroyed.
    @PreDestroy
    public void destroy() {
        System.out.println("Database connection closed");
    }

    // Method to simulate database connection work
    public void connect() {
        System.out.println("Connecting to the database...");
    }
}
```

Main.java

```
import
org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Main {

    public static void main(String[] args) {
        // Create Spring container using AnnotationConfigApplicationContext
        AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext();

        // Register the configuration class with Spring (in this case, we are
using a class annotated with @Component)
        context.scan("com.example.lifecycle");  // Change this to your package
where DatabaseConnection is located
        context.refresh();

        // Get the bean from Spring context
        DatabaseConnection dbConnection =
context.getBean(DatabaseConnection.class);

        // Simulate work
        dbConnection.connect();

        // Close the context manually
        context.close();
    }
}
```

3. Spring Boot Starter + Auto-Configuration + Properties:

GreetingService.java

```
import org.springframework.stereotype.Component;

@Component
public class GreetingService {

    @Value("${app.message}")
    private String msg;

    public String greet() {
        return msg != null ? msg : "Welcome to Spring Boot!";
    }
}
```

MainApplication.java

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

import javax.annotation.PostConstruct;
```

```java
@SpringBootApplication
public class MainApplication implements CommandLineRunner {

    @Autowired
    private GreetingService greetingService;

    @PostConstruct
    public void init() {
        System.out.println(greetingService.greet());
    }

    @Override
    public void run(String... args) throws Exception {
        // Any other logic can go here
    }

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

application.properties

```
server.port=9090
spring.main.banner-mode=off
app.message=Hello from properties!
```

pom.xml

```xml
<dependencies>
    <!-- Spring Boot Starter Web (for basic web functionality) -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!-- Spring Boot Starter (includes basic auto-configuration) -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter</artifactId>
    </dependency>

    <!-- Spring Boot Starter Test (for testing) -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

4. Console Level

GreetingService.java

```java
public class GreetingService {

    public String getMessage() {
        return "Hello from Spring Console App!";
    }
}
```

GreetingController.java

```
public class GreetingController {

    private GreetingService greetingService;

    // Constructor Injection
    public GreetingController(GreetingService greetingService) {
        this.greetingService = greetingService;
    }

    // Method to display the message
    public String showMessage() {
        return greetingService.getMessage();
    }
}
```

beans.xml

```
<?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-3.0.xsd">

    <!-- Declare the GreetingService bean -->
    <bean id="greetingService" class="GreetingService" />

    <!-- Declare the GreetingController bean, with constructor injection -->
    <bean id="greetingController" class="GreetingController">
        <constructor-arg ref="greetingService" />
    </bean>

</beans>
```

Main.java

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main {

    public static void main(String[] args) {
        // Load the Spring context from the beans.xml file
        ApplicationContext ctx = new
ClassPathXmlApplicationContext("beans.xml");

        // Get the GreetingController bean from the context
        GreetingController controller = ctx.getBean(GreetingController.class);

        // Display the message using the controller
        System.out.println(controller.showMessage());
    }
}
```

5. Spring Hibernate

Employee.java

```
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;

@Entity
```

```java
public class Employee {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String name;
    private double salary;

    // Getters and Setters
    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;
    }
}
```

hibernate.cfg.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration
DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>
        <!-- JDBC Database connection settings -->
        <property
name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/your_database</
property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">password</property>

        <!-- JDBC connection pool settings -->
        <property name="hibernate.c3p0.min_size">5</property>
        <property name="hibernate.c3p0.max_size">20</property>
        <property name="hibernate.c3p0.timeout">300</property>
        <property name="hibernate.c3p0.max_statements">50</property>
        <property name="hibernate.c3p0.idle_test_period">3000</property>

        <!-- Specify dialect -->
        <property
name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>

        <!-- Enable Hibernate's automatic session context management -->
        <property
```

```
name="hibernate.current_session_context_class">thread</property>

        <!-- Echo all executed SQL to stdout -->
        <property name="hibernate.show_sql">true</property>

        <!-- Drop and re-create the database schema on startup -->
        <property name="hibernate.hbm2ddl.auto">update</property>

        <!-- Mention annotated class -->
        <mapping class="Employee"/>
    </session-factory>
</hibernate-configuration>
```

EmployeeDAO.java

```
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
import java.util.List;

@Repository
public class EmployeeDAO {

    @Autowired
    private SessionFactory sessionFactory;

    public void saveEmployee(Employee employee) {
        Session session = sessionFactory.getCurrentSession();
        session.saveOrUpdate(employee);
    }

    public Employee getEmployeeById(int id) {
        Session session = sessionFactory.getCurrentSession();
        return session.get(Employee.class, id);
    }

    @SuppressWarnings("unchecked")
    public List<Employee> getAllEmployees() {
        Session session = sessionFactory.getCurrentSession();
        return session.createQuery("from Employee").list();
    }
}
```

applicationContext.xml

```
<?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-3.0.xsd">

    <!-- DataSource Configuration -->
    <bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSource">
        <property name="driverClassName" value="com.mysql.cj.jdbc.Driver" />
        <property name="url"
value="jdbc:mysql://localhost:3306/your_database" />
        <property name="username" value="root" />
        <property name="password" value="password" />
    </bean>

    <!-- Hibernate SessionFactory Configuration -->
    <bean id="sessionFactory"
class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="packagesToScan" value="com.example" />
        <property name="hibernateProperties">
            <props>
                <prop
key="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</prop>
                <prop key="hibernate.show_sql">true</prop>
                <prop key="hibernate.hbm2ddl.auto">update</prop>
            </props>
        </property>
    </bean>

    <!-- Hibernate Transaction Manager -->
    <bean id="transactionManager"
class="org.springframework.orm.hibernate5.HibernateTransactionManager">
        <property name="sessionFactory" ref="sessionFactory" />
    </bean>

    <!-- EmployeeDAO Bean Configuration -->
    <bean id="employeeDAO" class="com.example.EmployeeDAO" />

</beans>
```

Main.java

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.example.Employee;
import com.example.EmployeeDAO;

public class Main {

    public static void main(String[] args) {
        // Load the Spring application context
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

        // Get the DAO bean
        EmployeeDAO employeeDAO = context.getBean("employeeDAO",
EmployeeDAO.class);

        // Create and save a new employee
        Employee employee = new Employee();
        employee.setName("John Doe");
        employee.setSalary(50000);
        employeeDAO.saveEmployee(employee);

        // Retrieve and print all employees
        System.out.println("Employee List: ");
        for (Employee emp : employeeDAO.getAllEmployees()) {
            System.out.println("ID: " + emp.getId() + ", Name: " + emp.getName()
+ ", Salary: " + emp.getSalary());
        }
    }
}
```

pom.xml

```
<dependencies>
    <!-- Spring Framework -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.10</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-orm</artifactId>
        <version>5.3.10</version>
    </dependency>

    <!-- Hibernate -->
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>5.4.32.Final</version>
    </dependency>

    <!-- MySQL Driver -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.26</version>
    </dependency>
</dependencies>
```

6. Spring with Database(JDBC/JPA)

Student.java

```
public class Student {
    private int id;
    private String name;
    private String course;

    // Constructors
    public Student(int id, String name, String course) {
        this.id = id;
        this.name = name;
        this.course = course;
    }

    // Getters and Setters
    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 String getCourse() {
        return course;
    }

    public void setCourse(String course) {
        this.course = course;
    }
}
```

StudentDAO.java

```
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;

import javax.sql.DataSource;
import java.util.List;

public class StudentDAO {
    private JdbcTemplate jdbcTemplate;

    // Setter for JdbcTemplate
    public void setDataSource(DataSource dataSource) {
        this.jdbcTemplate = new JdbcTemplate(dataSource);
    }

    // Method to add a student
    public void addStudent(Student student) {
        String sql = "INSERT INTO student(id, name, course) VALUES(?, ?, ?)";
        jdbcTemplate.update(sql, student.getId(), student.getName(),
student.getCourse());
    }

    // Method to fetch all students
    public List<Student> getStudents() {
        String sql = "SELECT * FROM student";
        return jdbcTemplate.query(sql, new RowMapper<Student>() {
            @Override
            public Student mapRow(java.sql.ResultSet rs, int rowNum) throws
java.sql.SQLException {
                Student student = new Student(
                        rs.getInt("id"),
                        rs.getString("name"),
                        rs.getString("course")
                );
                return student;
            }
        });
    }
}
```

applicationContext.xml

```
<?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-4.3.xsd">

    <!-- DataSource Configuration for MySQL -->
    <bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="com.mysql.cj.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/springdb" />
        <property name="username" value="root" />
        <property name="password" value="password" />
    </bean>

    <!-- JdbcTemplate Bean -->
    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="dataSource" />
    </bean>
    <!-- StudentDAO Bean -->
    <bean id="studentDAO" class="StudentDAO">
        <property name="dataSource" ref="dataSource" />
    </bean>
</beans>
```

Main.java

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.util.List;

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

        // Get the DAO bean
        StudentDAO studentDAO = (StudentDAO) context.getBean("studentDAO");

        // Add a new student
        Student student1 = new Student(1, "John Doe", "Computer Science");
        studentDAO.addStudent(student1);

        // Fetch and display all students
        List<Student> students = studentDAO.getStudents();
        for (Student student : students) {
            System.out.println("ID: " + student.getId() + ", Name: " +
student.getName() + ", Course: " + student.getCourse());
        }
    }
}
```

pom.xml

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>5.3.10</version>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.23</version>
</dependency>
```