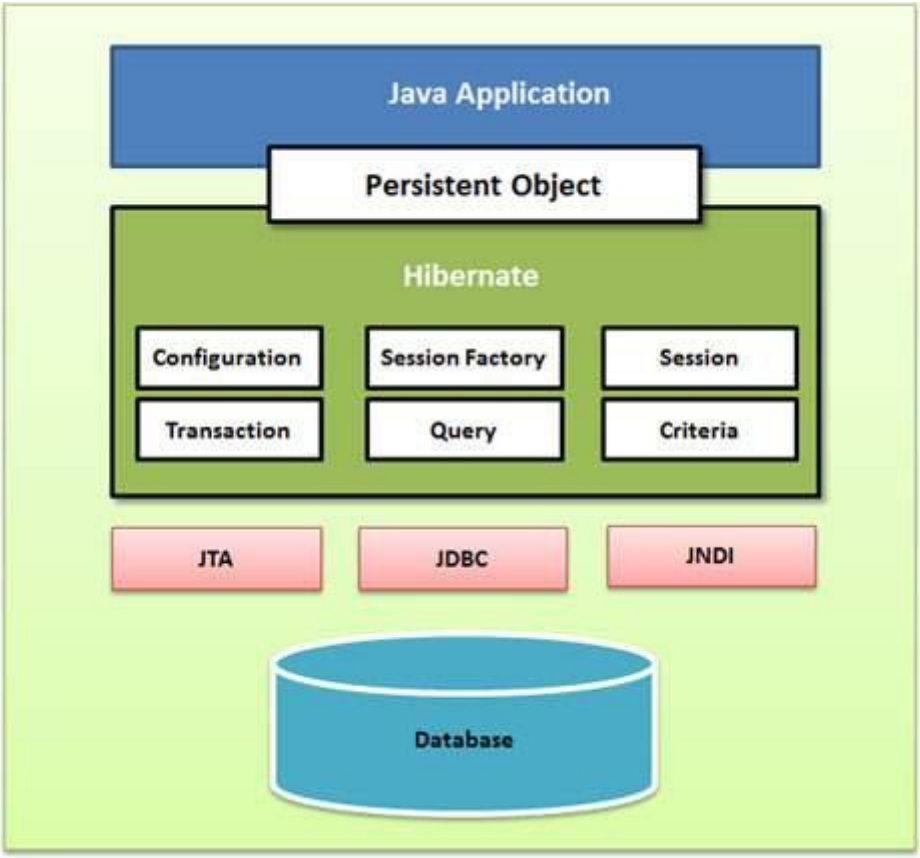
Prac 9) Write a program for implementing concept of Hibernate, Stuct, Spring

1) Hibernate :

Hibernate is an Object-Relational Mapping (ORM) solution for JAVA. It is an open source persistent framework created by Gavin King in 2001. It is a powerful, high performance Object-Relational Persistence and Query service for any Java Application.

Hibernate maps Java classes to database tables and from Java data types to SQL data types and relieves the developer from 95% of common data persistence related programming tasks.

Hibernate sits between traditional Java objects and database server to handle all the works in persisting those objects based on the appropriate O/R mechanisms and patterns.



1. Configuration Object

The Configuration object is the first Hibernate object you create in any Hibernate application. It is usually created only once during application initialization. It represents a configuration or properties file required by the Hibernate.

The Configuration object provides two keys components –

- **Database Connection** – This is handled through one or more configuration files supported by Hibernate. These files are **hibernate.properties** and **hibernate.cfg.xml**.
- **Class Mapping Setup** – This component creates the connection between the Java classes and database tables.

2. SessionFactory Object

SessionFactory object configures Hibernate for the application using the supplied configuration file and allows for a Session object to be instantiated. The SessionFactory is a thread safe object and used by all the threads of an application.

The SessionFactory is a heavyweight object; it is usually created during application start up and kept for later use. You would need one SessionFactory object per database using a separate configuration file. So, if you are using multiple databases, then you would have to create multiple SessionFactory objects.

3. Session Object

A Session is used to get a physical connection with a database. The Session object is lightweight and designed to be instantiated each time an interaction is needed with the database. Persistent objects are saved and retrieved through a Session object.

The session objects should not be kept open for a long time because they are not usually thread safe and they should be created and destroyed them as needed.

4. Transaction Object

A Transaction represents a unit of work with the database and most of the RDBMS supports transaction functionality. Transactions in Hibernate are handled by an underlying transaction manager and transaction (from JDBC or JTA).

5. Query Object

Query objects use SQL or Hibernate Query Language (HQL) string to retrieve data from the database and create objects. A Query instance is used to bind query parameters, limit the number of results returned by the query, and finally to execute the query.

6. Criteria Object

Criteria objects are used to create and execute object oriented criteria queries to retrieve objects.

Project Structure

src/main/java/

└── com.example/

    ├── Student.java

    ├── HibernateUtil.java

    └── App.java

resources/

└── hibernate.cfg.xml

pom.xml

Program Flow

1. **Program starts** in App.main()
2. Calls HibernateUtil.getSessionFactory()
   → reads hibernate.cfg.xml
   → connects to DB
   → builds SessionFactory
3. Opens a **Session** (like a DB connection)
4. Begins a **Transaction** (to group SQL operations)
5. Performs operations:

   - save() → INSERT SQL
   - get() → SELECT SQL
   - update() → UPDATE SQL
   - delete() → DELETE SQL

6. Commits transaction (saves changes)
7. Closes session.

2. Struts

Struts is a **Model–View–Controller (MVC)** framework for building Java EE web applications.
It helps you **separate concerns**:

- **Model** – Business logic, form beans
- **View** – JSP pages
- **Controller** – ActionServlet + Action classes

Struts is built on **Servlets + JSP**, but adds:

- A central **Controller** (`ActionServlet`)
- **XML configuration** for flow (`struts-config.xml`)
- **Action classes** to handle requests

Flow of Control

When user submits a form in Struts:

1 **User fills form** in `index.jsp`
2 On submit, request goes to `ActionServlet` via URL pattern `*.do`
3 `ActionServlet` checks **struts-config.xml**:

- Finds matching `<action>` mapping
- Loads `ActionForm` to hold form data
- Invokes mapped `Action` class
  4 `Action` class executes logic, sets result data in request
  5 Returns a `forward` name like `"success"`
  6 `ActionServlet` finds `<forward>` mapping and forwards to **view JSP**

3. Spring:

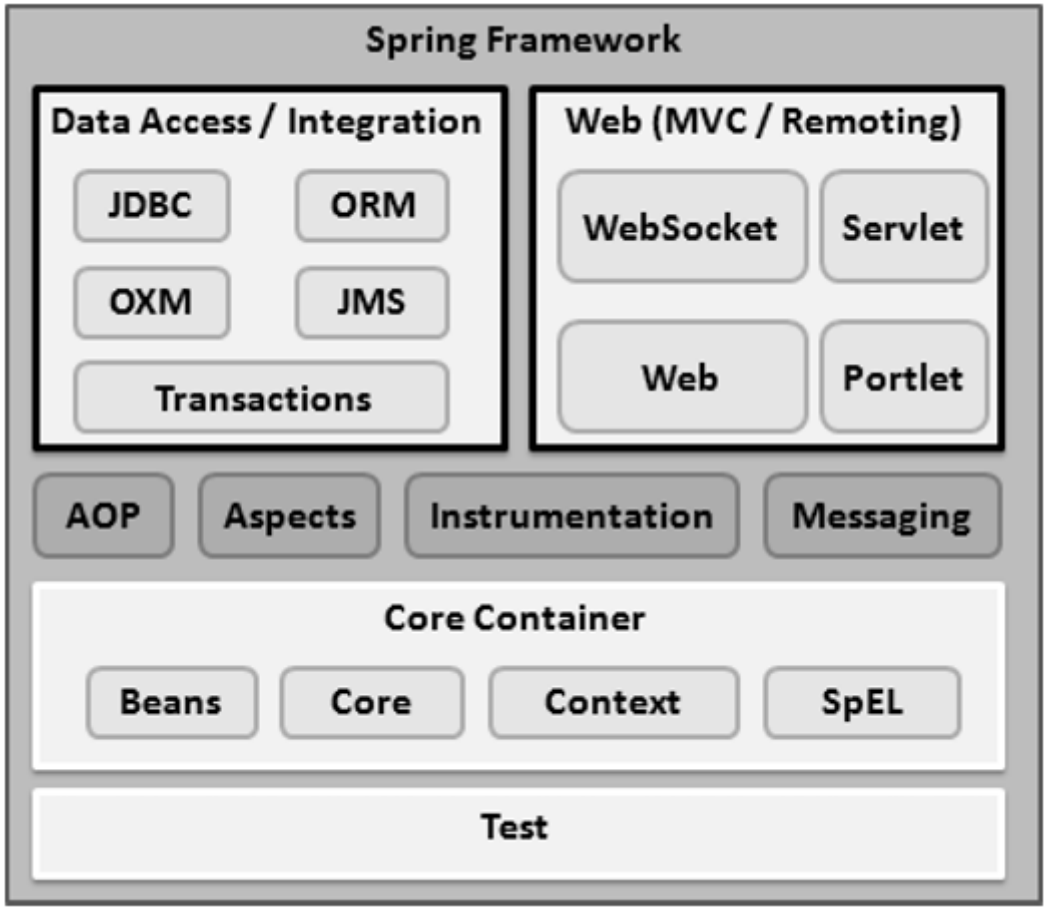
Spring framework is an open source Java platform. It was initially written by Rod Johnson and was first released under the Apache 2.0 license in June 2003.

Spring is lightweight when it comes to size and transparency. The basic version of Spring framework is around 2MB.

The core features of the Spring Framework can be used in developing any Java application, but there are extensions for building web applications on top of the Java EE platform. Spring framework targets to make J2EE development easier to use and promotes good programming practices by enabling a POJO-based programming model.

Spring Framework – Architecture :

The Spring Framework provides about 20 modules which can be used based on an application requirement.



Core Container

The Core Container consists of the Core, Beans, Context, and Expression Language modules the details of which are as follows –

- The **Core** module provides the fundamental parts of the framework, including the IoC and Dependency Injection features.
- The **Bean** module provides BeanFactory, which is a sophisticated implementation of the factory pattern.
- The **Context** module builds on the solid base provided by the Core and Beans modules and it is a medium to access any objects defined and configured. The ApplicationContext interface is the focal point of the Context module.
- The **SpEL** module provides a powerful expression language for querying and manipulating an object graph at runtime.

Data Access/Integration

The Data Access/Integration layer consists of the JDBC, ORM, OXM, JMS and Transaction modules whose detail is as follows –

- The **JDBC** module provides a JDBC-abstraction layer that removes the need for tedious JDBC related coding.

- The **ORM** module provides integration layers for popular object-relational mapping APIs, including JPA, JDO, Hibernate, and iBatis.
- The **OXM** module provides an abstraction layer that supports Object/XML mapping implementations for JAXB, Castor, XMLBeans, JiBX and XStream.
- The Java Messaging Service **JMS** module contains features for producing and consuming messages.
- The **Transaction** module supports programmatic and declarative transaction management for classes that implement special interfaces and for all your POJOs.

Web

The Web layer consists of the Web, Web-MVC, Web-Socket, and Web-Portlet modules the details of which are as follows –

- The **Web** module provides basic web-oriented integration features such as multipart file-upload functionality and the initialization of the IoC container using servlet listeners and a web-oriented application context.
- The **Web-MVC** module contains Spring's Model-View-Controller (MVC) implementation for web applications.
- The **Web-Socket** module provides support for WebSocket-based, two-way communication between the client and the server in web applications.
- The **Web-Portlet** module provides the MVC implementation to be used in a portlet environment and mirrors the functionality of Web-Servlet module.

Miscellaneous

There are few other important modules like AOP, Aspects, Instrumentation, Web and Test modules the details of which are as follows –

- The **AOP** module provides an aspect-oriented programming implementation allowing you to define method-interceptors and pointcuts to cleanly decouple code that implements functionality that should be separated.
- The **Aspects** module provides integration with AspectJ, which is again a powerful and mature AOP framework.
- The **Instrumentation** module provides class instrumentation support and class loader implementations to be used in certain application servers.
- The **Messaging** module provides support for STOMP as the WebSocket sub-protocol to use in applications. It also supports an annotation programming model for routing and processing STOMP messages from WebSocket clients.
- The **Test** module supports the testing of Spring components with JUnit or TestNG frameworks.