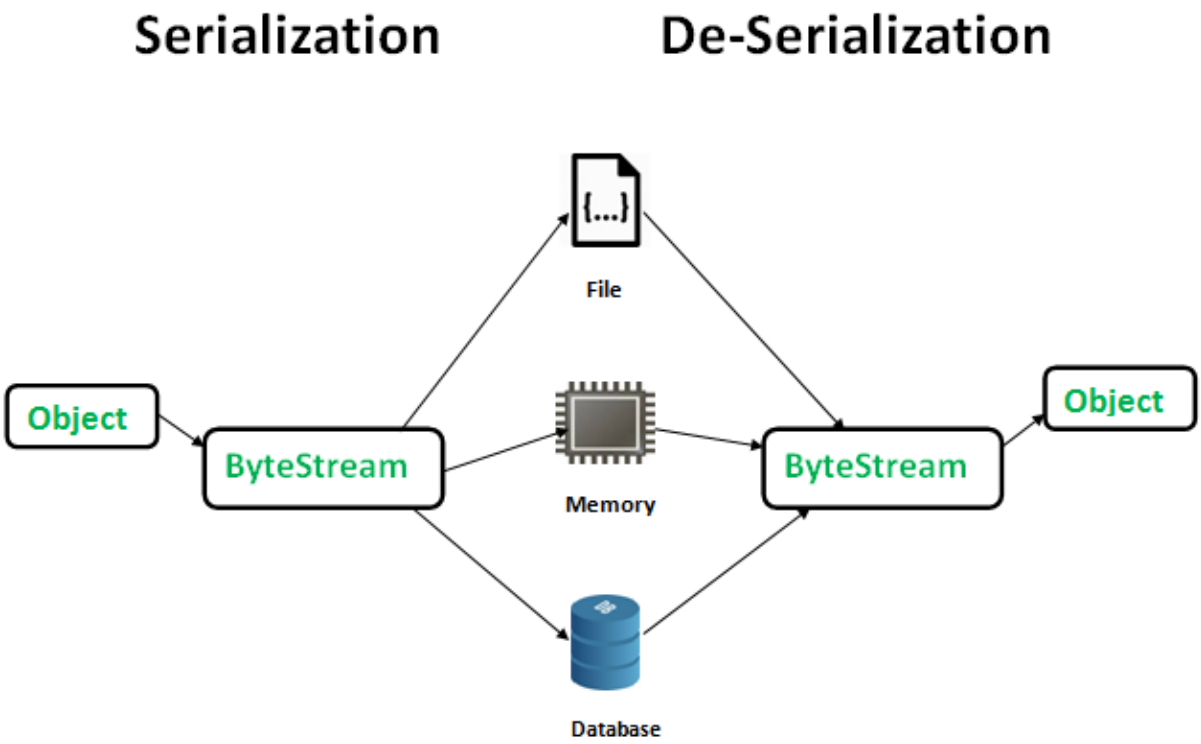
Important Question and Answer for End-Term- Ms.Ramabai V
Outline Serialization and Deserialization
Serialization is a mechanism of converting the state of an object into a byte stream. Deserialization is the reverse process where the byte stream is used to recreate the actual Java object in memory. This mechanism is used to persist objects.



The byte stream created is platform independent. So, the object serialized on one platform can be deserialized on a different platform. To make a Java object serializable we implement the **java.io.Serializable** interface. The ObjectOutputStream class contains **writeObject()** method for serializing an Object.

Advantages of Serialization

1. To save/persist state of an object.
2. To travel an object across a network.

Define Servlet

Servlet technology is used to create a web application (resides at server side and generates a dynamic web page).

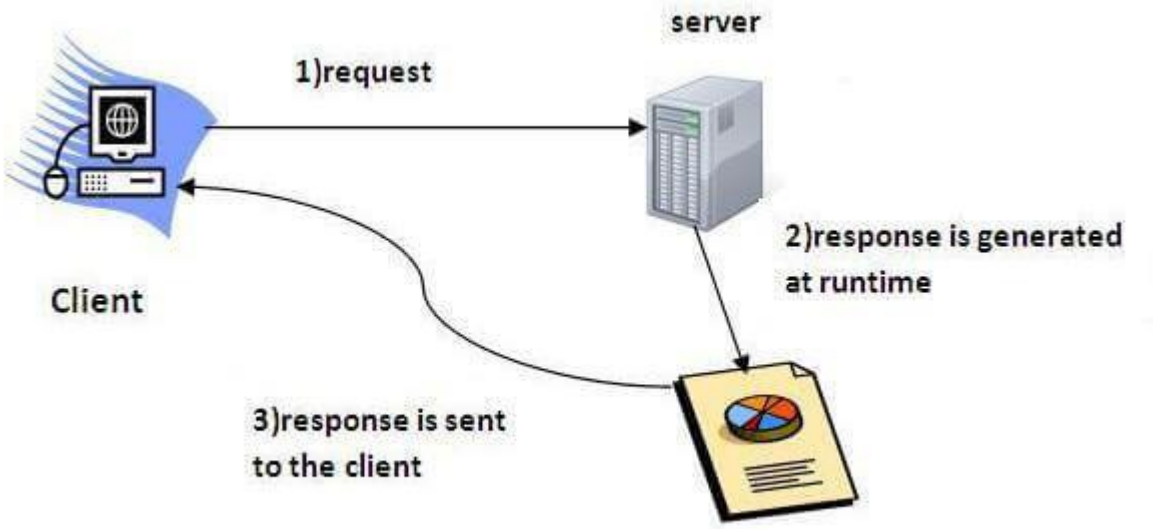
Servlet technology is robust and scalable because of java language. Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language. However, there were many disadvantages to this technology.

There are many interfaces and classes in the Servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse, etc.

What is a Servlet?

Servlet can be described in many ways, depending on the context.

- Servlet is a technology which is used to create a web application.
- Servlet is an API that provides many interfaces and classes including documentation.
- Servlet is an interface that must be implemented for creating any Servlet.
- Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
- Servlet is a web component that is deployed on the server to create a dynamic web page.



List the three types of scripting elements in Jsp.
You use JSP scripting elements to include Java code within a JSP. There are three types of scripting elements: expressions, scriptlets, and declarations.
SP Scripting elements

The scripting elements provides the ability to insert java code inside the jsp. There are three types of scripting elements:

- scriptlet tag
- expression tag
- declaration tag

JSP scriptlet tag

A scriptlet tag is used to execute java source code in JSP. Syntax is as follows:

1. <% java source code %>

The code placed within **JSP expression tag** is *written to the output stream of the response*. So you need not write out.print() to write data. It is mainly used to print the values of variable or method.

Syntax of JSP expression tag

1. <%= statement %>

JSP Declaration Tag

The **JSP declaration tag** is used *to declare fields and methods*.

The code written inside the jsp declaration tag is placed outside the service() method of auto generated servlet.

So it doesn't get memory at each request.

Syntax of JSP declaration tag

The syntax of the declaration tag is as follows:

1. <%! field or method declaration %>

Define JPQL.

The JPQL (Java Persistence Query Language) is an object-oriented query language which is used to perform database operations on persistent entities. Instead of database table, JPQL uses entity object model to operate the SQL queries. Here, the role of JPA is to transform JPQL into SQL. Thus, it provides an easy platform for developers to handle SQL tasks.

JPQL is an extension of Entity JavaBeans Query Language (EJBQL), adding the following important features to it: -

- It can perform join operations.
- It can update and delete data in a bulk.
- It can perform aggregate function with sorting and grouping clauses.
- Single and multiple value result types.

JPQL Features

- It is a platform-independent query language.
- It is simple and robust.
- It can be used with any type of database such as MySQL, Oracle.
- JPQL queries can be declared statically into metadata or can also be dynamically built in code.

Creating Queries in JPQL

JPQL provides two methods that can be used to access database records. These methods are: -

- Query createQuery(String name) - The createQuery() method of EntityManager interface is used to create an instance of Query interface for executing JPQL statement.

1. **Query query = em.createQuery("Select s.s_name from StudentEntity s");**

This method creates dynamic queries that can be defined within business logic.

- Query createNamedQuery(String name) - The createNamedQuery() method of EntityManager interface is used to create an instance of Query interface for executing named queries.

1. **@NamedQuery(name = "find name" , query = "Select s from StudentEntity s")**

This method is used to create static queries that can be defined in entity class.

Now, we can control the execution of query by the following Query interface methods: -

- **int executeUpdate()** - This method executes the update and delete operation.
- **int getFirstResult()** - This method returns the first positioned result the query object was set to retrieve.
- **int getMaxResults()** - This method returns the maximum number of results the query object was set to retrieve.
- **java.util.List getResultList()** - This method returns the list of results as an untyped list.
- **Query setFirstResult(int startPosition)** - This method assigns the position of first result to retrieve.
- **Query setMaxResults(int maxResult)** - This method assigns the maximum numbers of result to retrieve.

Outline the benefits of hibernate framework.

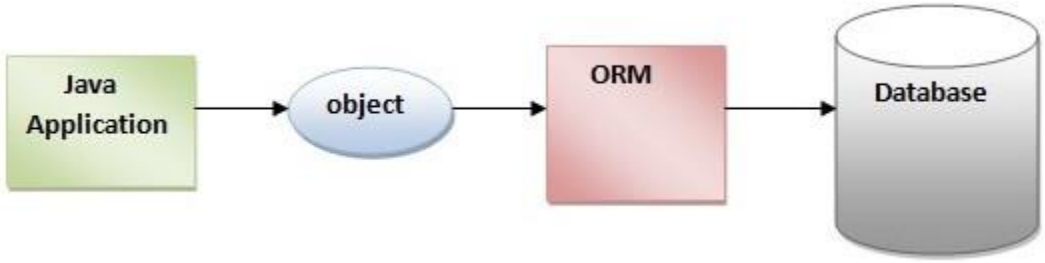
This hibernate tutorial provides in-depth concepts of Hibernate Framework with simplified examples. It was started in 2001 by Gavin King as an alternative to EJB2 style entity bean.

Hibernate Framework

Hibernate is a Java framework that simplifies the development of Java application to interact with the database. It is an open source, lightweight, ORM (Object Relational Mapping) tool. Hibernate implements the specifications of JPA (Java Persistence API) for data persistence.

ORM Tool

An ORM tool simplifies the data creation, data manipulation and data access. It is a programming technique that maps the object to the data stored in the database.



The ORM tool internally uses the JDBC API to interact with the database.

What is JPA?

Java Persistence API (JPA) is a Java specification that provides certain functionality and standard to ORM tools. The **javax.persistence** package contains the JPA classes and interfaces.

Advantages of Hibernate Framework

Following are the advantages of hibernate framework:

1) Open Source and Lightweight

Hibernate framework is open source under the LGPL license and lightweight.

2) Fast Performance

The performance of hibernate framework is fast because cache is internally used in hibernate framework. There are two types of cache in hibernate framework first level cache and second level cache. First level cache is enabled by default.

3) Database Independent Query

HQL (Hibernate Query Language) is the object-oriented version of SQL. It generates the database independent queries. So you don't need to write database specific queries. Before Hibernate, if database is changed for the project, we need to change the SQL query as well that leads to the maintenance problem.

4) Automatic Table Creation

Hibernate framework provides the facility to create the tables of the database automatically. So there is no need to create tables in the database manually.

5) Simplifies Complex Join

Fetching data from multiple tables is easy in hibernate framework.

6) Provides Query Statistics and Database Status

Hibernate supports Query cache and provide statistics about query and database status

Illustrate the advantages of Spring framework.

Spring framework helps develop various types of applications using the Java platforms. It provides an extensive level of infrastructure support. Spring also provides the "Plain Old Java Objects" (POJOs) mechanisms using which developers can easily create the Java SE programming model with the full and partial JAVA EE (Enterprise Edition).

Spring strives to facilitate the complex and unmanageable enterprise Java application development revolution by offering a framework that incorporates technologies, such as:

- Aspect-oriented Programming (AOP)
- Dependency Injection (DI)
- Plain Old Java Object (POJO)

Spring framework provides plenty of features. It helps application developers to perform the following functions:

- Create a Java method that runs in a database transaction with no help from transaction APIs.
- Create a local Java method that defines a remote procedure with no help from remote APIs.
- Create a local Java method for a management operation with no help from JMX APIs.
- Create a local Java method for a message handler with no help from JMS APIs.

Spring is a lightweight framework. It provides the best mechanisms for different frameworks, such as Struts, Hibernate, EJB, JSF, and Tapestry. It helps solve real-time technical problems. Spring contains multiple modules, such as WEB MVC, IOC, DAO, AOP, Context, and ORM.

Spring also helps create scalable, secure, and robust business-based web applications. We can consider the Spring framework a cluster of sub frameworks such as Spring Web Flow, Spring ORM, and Spring MVC. In expansion to Java, Spring also sustains Kotlin and Groovy.

The Spring framework provides a base that controls all the other Spring-based projects, such as:

1. Springboot
2. Spring Cloud
3. Spring GraphQL

Spring Framework in Java: Advantages

Using the Spring framework, developers can leverage the below-listed advantages:

- **Pre-defined Templates**

Spring framework contains various types of templates for Hibernate, JDBC, and JPA technologies. With the help of this approach, developers are not required to define complex code.

Example: JdbcTemplate - Here, we do not need to write the logic for creating a statement, committing the transaction, creating a connection, and exception handling. It saves the time-consuming approach.

- **Loose Coupling**

We can consider Spring applications to be loosely coupled as per the dependency injection mechanisms.

- **Easy and Simple to Test**

It is easy to test the entire application using a spring framework with a dependency injection mechanism. The EJB or Struts application requires the server to execute the application.

- **Non-invasive**

As per the Plain Old Java Object (POJO) technique, Spring is easy to implement as it does not force the developer to inherit certain classes or implementations on any interface.

- **Fast Development**

With the help of Dependency Injection, it is easy to integrate the framework and support the development of JavaEE-based applications.

- **Strong Abstraction Support**

Spring supports the strong abstraction capability for Java EE-based specifications, such as JMS, JDBC, JPA, and JTA.

- **Spring's Web Framework is Well-Organized**

It is a web MVC framework that delivers a fantastic option to web frameworks for developing applications using Struts or different widespread web frameworks.

- **Spring Delivers a Suitable API**

It translates technology-specific anomalies thrown by JDBC, Hibernate, or JDO into uniform, uncontrolled exceptions.

- **Lightweight IoC**

It is lightweight, particularly when compared to EJB containers, for example. This helps create and deploy applications on computers with restricted memory and CPU resources.

- **Constant Transaction Management**

Spring provides an interface that can help scale down to a local transaction (for example, using a single database) and scale up to global transactions (for example, JTA).

Spring Core

In the Spring framework, we have certain features as discussed below:

- **Dependency Injection (DI)**

Dependency Injection is the core of Spring Framework. We can define the concept of Spring with the Inversion of Control (IoC). DI allows the creation of dependent objects outside of a class and provides those objects to a class in different ways. Dependency Injection can be utilized while defining the parameters to the Constructor or by post-construction using Setter methods.

The dependency feature can be summarized into an association between two classes. For example, suppose class X is dependent on class Y. Now, it can create many problems in the real world, including system failure. Hence such dependencies need to be avoided. Spring IOC resolves such dependencies with Dependency Injection. Here, it indicates that class Y will get injected into class X by the IoC. DI thus makes the code easier to test and reuse.

While creating a complex Java application, application classes should be independent of other Java classes to improve the possibility of reusing these classes and to test them independently of other classes during unit testing. Dependency Injection enables these classes to be together, and at the same time, keeps them independent.

- **Support for Aspect-Oriented Programming**

AOP provides more modularity to the cross-cutting challenges in applications.

Here are the functions we can use in our applications as per certain real-time challenges:

1. Logging
2. Caching
3. Transaction management
4. Authentication

AOP has the in-built object-oriented programming capabilities to define the structure of the program, where OOP modularity is established in classes.

In AOP, the primary unit of modularity is a factor (cross-cutting concern). This allows users to use AOP to build custom aspects and declarative enterprise services. The IoC container does not depend on AOP; it provides the custom enabled based capabilities which allow writing logic as per the programming method.

However, Aspect-Oriented Programming integrated with the Spring IoC delivers a robust middleware solution.

- **Data Access Framework**

Database communication problems are one of the common challenges which developers encounter when creating applications. Spring facilitates the database communication strategy by delivering immediate support for widespread data access frameworks in Java, such as Hibernate, JDBC, and Java Persistence API (JPA).

Additionally, it suggests resource management, exception handling, and resource wrapping for all the supported data access frameworks, further streamlining the development revolution.

- **Transaction Management Framework**

Java Transaction API (JTA), the Spring Transaction Management Framework, is not restricted to nested and global types of transactions. Spring presents an abstraction mechanism for Java that permits users to:

1. Work with local, international, and nested transactions wise logics
2. Savepoints
3. Simplify transaction management across the application

The Spring Data Access Framework instantly combines with the Transaction Management Framework with help for messaging and caching. This allows developers to build feature-rich transactional systems that span across the applications without relying on EJB or JTA.

- **Spring MVC Framework**

The Spring MVC allows developers to develop applications utilizing the popular MVC pattern. It is a request-based framework that enables developers to develop custom MVC implementations that efficiently serve their needs.

The core component of Spring MVC is the DispatcherServlet class, which manages user requests and then delivers them to the right controller. This permits the controller to process the request, create the model, and then deliver the data to the end-user via a restricted view.

- **Spring Web Service**

This Spring Web Service component supplies a streamlined way to build and handle web service endpoints in the application. It delivers a layered approach that can be controlled using XML. It can also be used to deliver mapping for web requests to a specific object.

- **Spring Test Frameworks**

Testing is a key component of application development. Spring streamlines testing within the framework with components like:

1. Mock objects
2. TestContext framework
3. Spring MVC Test

- **Core Container**

This includes the essential modules that are the cornerstone of the Spring framework.

- Core (spring-core) is the framework's core that controls features such as Inversion of Control and dependency injection.
- Beans (spring-beans) deliver BeanFactory, an advanced execution of the factory pattern.

- Context (spring-context) produces on Core and Beans and delivers a medium to access restricted objects. ApplicationContext interface is the core part of the Context module, and the spring-context support provides help for third-party interactions such as caching, mailing, and template engines.
- SpEL (spring-expression) allows users to use the Spring Expression Language to query and control the object graph at execution time.

- **Data Access/Integration**

This contains the modules used to manage data access and transaction processing in an application.

- JDBC (spring-jdbc) delivers a JDBC abstraction layer that eliminates the need to split JDBC coding when dealing using databases.
- ORM (spring-orm) are essential integration layers for overall object-relational mapping API, for example, JDO Hibernate, JPA, etc.
- OXM (spring-oxm) is the abstraction layer that supports Object/XML mapping implementations, for example, JAXB, XStream, etc.
- JMS (spring-jms) is the Java Messaging Service module that constructs and consumes messages that instantly incorporate the Spring messaging module.
- Transaction (spring-tx) offers programmatic and declarative transaction management for classes that include unique interfaces and POJOs.

8. State the two types of tools using which Spring projects can be created?
Gradle and Maven

9. What is Selenium?

Selenium is a free (open-source) automated testing framework used to validate web applications across different browsers and platforms. You can use multiple programming languages like Java, C#, Python, etc to create Selenium Test Scripts. Testing done using the Selenium testing tool is usually referred to as **Selenium Testing**.

10. What does persistence.xml files contain?

```
<?xml version="1.0" encoding="UTF-8"?>

<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="2.2"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_2.xsd">

<persistence-unit name="mypersistence" transaction-
type="RESOURCE_LOCAL">
```

```
<provider>org.hibernate.jpa.HibernatePersistenceProvider</prov
ider>

<properties>

<property name="javax.persistence.jdbc.driver"
value="com.mysql.jdbc.Driver"/>

<property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost/JAVAFS"/>

<property name="javax.persistence.jdbc.user" value="root"/>

<property name="javax.persistence.jdbc.password" value=""/>

<property name="hibernate.dialect"
value="org.hibernate.dialect.MySQL8Dialect"/>

<property name="hibernate.show_sql" value="true"/>

<property name="hibernate.format_sql" value="true"/>

<property name="hibernate.hbm2ddl.auto" value="update"/>

</properties>

</persistence-unit>

</persistence>
```

PART B

ANSWER ALL THE QUESTIONS (4 X 5 = 20M)

11. What is ORM and JPA,Name Some ORM? Write a program to save a product using ORM?

12. Explain the traversing methods of Hash Map? [CO1,Comprehension]

13. Explain the Servlet Life Cyle and its architecture.

14. Expalin One to Many Relationship of Object using Orm Framework and JPA?

15 Describe Spring MVC and its Architecture?

Client→ Controller→ Service→Dao→ Database

Dao depends on Database Spring does dependency Injection of Database to Dao.

Dao has interface and implementation.

Service depends on Dao ,Spring creates the dependency object of Dao and injects it to Service, using Autowired annotation.

Service has interface and implementation.

Controller depends on Service, Spring does the dependency Injection

Controller doesn't have interface, because it process request and response

16. Write a program for Dependency Injection Take Book and Author as usecases?
//book

```
 package com.abc.entity;

public class Book {

private int bookId;

private String bookName;

private float bookPrice;

private Author author;

public int getBookId() {

return bookId;

}

public void setBookId(int bookId) {

this.bookId = bookId;

}

public String getBookName() {

return bookName;

}

public void setBookName(String bookName) {

this.bookName = bookName;

}

public float getBookPrice() {

return bookPrice;

}

public void setBookPrice(float bookPrice) {

this.bookPrice = bookPrice;

}
```

```
public Author getAuthor() {
return author;
}
public void setAuthor(Author author) {
this.author = author;
}
}
//Author
package com.abc.bean;
public class Author {
private String authorName;
private String authorMailid;
public String getAuthorName() {
return authorName;
}
public void setAuthorName(String authorName) {
this.authorName = authorName;
}
public String getAuthorMailid() {
return authorMailid;
}
public void setAuthorMailid(String authorMailid) {
this.authorMailid = authorMailid;
}
}
```

```
//BookAuthorMain

package com.abc.main;



import org.springframework.context.ApplicationContext;

import
org.springframework.context.support.ClassPathXmlApplicationCon
text;



import com.abc.bean.Book;



public class BookAuthormain {



     public static void main(String[] args) {




          ApplicationContext context= new
ClassPathXmlApplicationContext("springconfig8.xml");

     Book b    =(Book) context.getBean("book");

     System.out.println("bookId"+b.getBookId());

     System.out.println("bookName"+b.getBookName());

     System.out.println("bookPrice"+b.getBookPrice());

     System.out.println("AuthorName"+b.getAuthor().getAuthorNa
me());

     System.out.println("AuthorMailId"+b.getAuthor().getAuthor
Mailid());



     }
```

```
}

// springconfig8.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

https://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="hello" class="com.abc.bean.HelloWorld">

<property name="message"

value="hellobatch8 listen to the class" />

</bean>

<bean id="book" class="com.abc.bean.Book">

<property name="bookId" value="1" />

<property name="bookName" value="coding in JFS" />

<property name="bookPrice" value="100" />

<property name="author" ref="author" />

</bean>

<bean id="author" class="com.abc.bean.Author">

<property name="AuthorName" value="Ramabai" />

<property name="AuthorMailid" value="rama@gmail.com" />

</bean>

</beans>
```

Write a program to fetch product details using hibernate ORM?

```
package com.abc.main;
```

```
import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

import javax.persistence.Persistence;

import com.abc.entity.Product;

public class FetchProduct {

public static void main(String[] args) {

EntityManagerFactory
emf=Persistence.createEntityManagerFactory("mypersistence");

EntityManager em=emf.createEntityManager();

int productId=101;

Product product=em.find(Product.class, productId);

if(product!=null)

{

System.out.println("ProductName:"+product.getProductName());

System.out.println("ProductPrice"+product.getProductPrice());

System.out.println("ProductcreatedOn"+product.getCreatedOn());

}

else

{

System.out.println("Product doesn't exist");

}

}

}
```

5.What is dependency Injection? Advantage of loose coupling?

6.Structure of hibernate Framework

7.What are the dependency software downloaded from maven repository for spring?

```
/ https://mvnrepository.com/artifact/org.springframework/spring-context

implementation group: 'org.springframework', name: 'spring-context', version: '5.3.24'
```

7.What are the dependency software downloaded from maven repository for mysql jdbc?

```
// https://mvnrepository.com/artifact/mysql/mysql-connector-java

implementation group: 'mysql', name: 'mysql-connector-java', version: '8.0.33'
```

8.What are the dependency software for spring core, spring jpa, mysql jdbc and hibernate orm?

mysql jdbc:

```
// https://mvnrepository.com/artifact/mysql/mysql-connector-java

implementation group: 'mysql', name: 'mysql-connector-java', version: '8.0.33'
```

hibernate orm:

```
// https://mvnrepository.com/artifact/org.hibernate/hibernate-core

implementation group: 'org.hibernate', name: 'hibernate-core', version: '5.6.14.Final'
```

spring core

```
// https://mvnrepository.com/artifact/org.springframework/spring-context

implementation group: 'org.springframework', name: 'spring-context', version: '5.3.24'
```

spring jpa

```
// https://mvnrepository.com/artifact/org.springframework/spring-orm

implementation group: 'org.springframework', name: 'spring-orm', version: '5.3.24'
```

9.Show view how spring works through a diagram?

Ypur application classes combined with configuration metadata so that ,after the application context is created and initialized,you have fully configured and executable system or application.

Refer this for the diagram

https://docs.spring.io/spring-framework/reference/core/beans/basics.html

10.Write the configuration meta-data for Spring Framework using any usecase?

```
<?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

https://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="book" class="com.abc.bean.Book">

<property name="bookId" value="21" />

<property name="bookName" value="coding in JFS" />

<property name="bookPrice" value="100" />

<property name="author" ref="author" />

</bean>

<bean id="author" class="com.abc.bean.Author">

<property name="authorName" value="Ramabai" />

<property name="authorMailid" value="rama@gmail.com" />

</bean>

</beans>
```

Part C

1.Servlet add and Database program?

2. JSP add and Database program?

3. MVC program