Part - A

1. **What is spring?**
   The Spring Framework is an open-source framework for building enterprise applications in Java. It provides comprehensive infrastructure support and is known for its modularity, flexibility, and scalability. The Spring Framework simplifies the development of complex, large-scale applications by providing a set of reusable and loosely coupled components.

2. **What is POJO?**
   POJO stands for Plain Old Java Object. It is an ordinary Java object, not bound by any special restriction other than those forced by the Java Language Specification and not requiring any classpath. POJOs are used for increasing the readability and re-usability of a program. The benefit of using only POJOs is that you do not need an EJB container product such as an application server but you have the option of using only a robust servlet container such as Tomcat or some commercial product.

3. **What is IoC?**
   Spring IoC (Inversion of Control) Container is the core of Spring Framework. It creates the objects, configures and assembles their dependencies, manages their entire life cycle. The Container uses Dependency Injection (DI) to manage the components that make up the application.

4. **What is AOP?**
   Aspect-oriented Programming (AOP) complements Object-oriented Programming (OOP) by providing another way of thinking about program structure. The key unit of modularity in OOP is the class, whereas in AOP the unit of modularity is the aspect

5. **What is Spring Security?**
   Spring Security is a powerful and highly customizable security framework that provides authentication, authorization, and other security features for Spring-based applications. It is a widely used open-source project that helps developers to secure their web applications by implementing security policies and rules.

6. **What are java beans?**
   Java Beans are special type of Pojos. JavaBeans are classes which encapsulate several objects into a single object. It helps in accessing these objects from multiple places. JavaBeans contains several elements like Constructors, Getter/Setter Methods and much more.

7. **What is Dependency Injection?**
   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.

Part - B

1. **Explain the application of Spring.**

Applications of Spring:

1) **POJO Based:**
   Spring enables developers to develop enterprise-class applications using POJOs. The benefit of using only POJOs is that you do not need an EJB container product such as an application server but you have the option of using only a robust servlet container such as Tomcat or some commercial product.

2) **Modular:**
   Spring is organized in a modular fashion. Even though the number of packages and classes are substantial, you have to worry only about the ones you need and ignore the rest.

3) **Integration with existing frameworks:**
   Spring does not reinvent the wheel, instead it truly makes use of some of the existing technologies like several ORM frameworks, logging frameworks, JEE, Quartz and JDK timers, and other view technologies.

4) **Testability:**
   Testing an application written with Spring is simple because environment-dependent code is moved into this framework. Furthermore, by using JavaBean style POJOs, it becomes easier to use dependency injection for injecting test data.

5) **Web MVC:**
   Spring's web framework is a well-designed web MVC framework, which provides a great alternative to web frameworks such as Struts or other over-engineered or less popular web frameworks.

6) **Central Exception Handling**:
   Spring provides a convenient API to translate technology-specific exceptions (thrown by JDBC, Hibernate, or JDO, for example) into consistent, unchecked exceptions.

7) **Lightweight:**
   IoC containers tend to be lightweight, especially when compared to EJB containers, for example. This is beneficial for developing and deploying applications on computers with limited memory and CPU resources.

8) **Transaction management**:
   Spring provides a consistent transaction management interface that can scale down to a local transaction (using a single database, for example) and scale up to global transactions (using JTA, for example).

2. **Write short notes about the AOP Terminologies.**

Aspect-oriented Programming (AOP) complements Object-oriented Programming (OOP) by providing another way of thinking about program structure. The key unit of modularity in OOP is the class, whereas in AOP the unit of modularity is the aspect.

- One of the key components of Spring is the AOP framework. While the Spring IoC container does not depend on AOP, it complements Spring IoC to provide a very capable middleware solution. It accomplishes this task by adding extra behaviour to existing code without modifying and changing the code of the software. We can declare the new code and the new behaviours separately as per our software requirements. Spring's AOP framework helps us to implement these cross-cutting concerns for our web applications.
- Aspect-Oriented Programming entails breaking down program logic into distinct parts called so-called concerns. The functions that span multiple points of an application are called cross-cutting concerns and these cross-cutting concerns are conceptually separate from the application's business logic.
- There are various common good examples of aspects like logging, auditing, declarative transactions, security, caching, etc.
- Dependency Injection helps you decouple your application objects from each other and AOP helps you decouple cross-cutting concerns from the objects that they affect. Spring AOP module provides interceptors to intercept an application. For example, when a method is executed, you can add extra functionality before or after the method execution

3. **Write short notes about the Spring ORM technique.**

Spring-ORM is a technique or a Design Pattern used to access a relational database from an object-oriented language. ORM (Object Relation Mapping) covers many persistence technologies. They are as follows:

JPA(Java Persistence API):

It is mainly used to persist data between Java objects and relational databases. It acts as a bridge between objectoriented domain models and relational database systems.

JDO(Java Data Objects):

It is one of the standard ways to access persistent data in databases, by using plain old Java objects (POJO) to represent persistent data.

Hibernate: It is a Java framework that simplifies the development of Java applications to interact with the database.

Oracle Toplink, and iBATIS: Oracle TopLink is a mapping and persistence framework for Java development.

For the above technologies, Spring provides integration classes so that each of these techniques can be used following Spring principles of configuration, and easily integrates with Spring transaction management. For each of the above technologies, the configuration consists of injecting the DataSource bean into some kind of SessionFactory or EntityManagerFactory, etc. For pure JDBC (Java Database Connectivity), integration classes are not required apart from JdbcTemplate because JDBC only depends on a DataSource. If someone wants to use an ORM like JPA (Java Persistence API) or Hibernate then you do not need spring-JDBC, but only this module.

4. Explain the Programmatic vs Declarative Transactions.

Spring supports two types of transaction management:

- **Programmatic transaction management**:
  This means that you have to manage the transaction with the help of programming. That gives you extreme flexibility, but it is difficult to maintain.
- **Declarative transaction management**:
  This means you separate transaction management from the business code. You only use annotations or XML-based configuration to manage the transactions.

Declarative transaction management is preferable over programmatic transaction management though it is less flexible than programmatic transaction management, which allows you to control transactions through your code. But as a kind of crosscutting concern, declarative transaction management can be modularized with the AOP approach. Spring supports declarative transaction management through the Spring AOP framework.

5. Describe the Spring Remoting technologies.

Spring has integration classes for remoting support that use a variety of technologies. The Spring framework simplifies the development of remoteenabled services. It saves a significant amount of code by having its own API. The remote support simplifies the building of remote-enabled services, which are implemented by your standard (Spring) POJOs. Spring now supports the following remoting technologies:

1. Remote Method Invocation (RMI)
2. Hessian
3. Burlap
4. Spring's HTTP invoker
5. JAX-RPC
6. JAX-WS
7. JMS

1) Exposing services using RMI

Transparently expose your services across the RMI infrastructure by using Spring's RMI support. After you've done this, you'll have a setup that's similar to remote EJBs, except there's no standard support for security context propagation or remote transaction propagation. When utilizing the RMI invoker, Spring provides hooks for such extra invocation context, so you may,

for example, plug-in security frameworks or custom security credentials.

- Using the RmiServiceExporter to export the service.
- Client-side service integration.

2) **Using Hessian to call services remotely through HTTP**

Spring has its own remoting service that supports HTTP serialization. HTTP Invoker makes use of the classes HttpInvokerServiceExporter and HttpInvokerProxyFactoryBean.

- Configuring the DispatcherServlet for Hessian and company.
- Using the HessianServiceExporter to expose your beans
- Connecting the client to the service

3) **Using Burlap**

It is the same as Hessian but XML-based implementation provided by Coucho. The classes used in Burlap are BurlapServiceExporter and BurlapProxyFactoryBean.

4) **Using HTTP invokers to expose services**

Spring HTTP invokers employ the regular Java serialization technique to expose services through HTTP, as opposed to Burlap and Hessian, which are both lightweight protocols with their own compact serialization mechanisms. This is especially useful if your arguments and return types are complicated and cannot be serialized using the serialization procedures used by Hessian and Burlap (refer to the next section for more considerations when choosing a remoting technology).

- Displaying the service object
- Client-side service integration

5) **Exposing servlet-based web services using JAX-RPC**

Spring provides a convenience base class for JAX-RPC servlet endpoint implementations – ServletEndpointSupport. To expose our AccountService we extend Spring's ServletEndpointSupport class and implement our business logic here, usually delegating the call to the business layer.

- Using JAX-RPC to gain access to online services
- JAX-RPC Bean Mappings Registration
- Adding your own JAX-RPC Handler

6) **Using JAX-WS to provide servlet-based web services**

SpringBeanAutowiringSupport is a useful basic class for JAX-WS servlet endpoint implementations. We modify Spring's SpringBeanAutowiringSupport class to expose our AccountService and execute our business logic here, generally outsourcing the request to the business layer.

7) **JMS**

Using JMS (Java Message Service) as the underlying communication protocol, it is also feasible to provide services transparently. The Spring Framework's JMS remoting

functionality is fairly minimal – it transmits and receives on the same thread and in the same non-transactional Session, thus performance will be very implementation dependant. It's worth noting that these single threaded and non-transactional restrictions only apply to Spring's JMS remoting functionality.

6. Explain the Spring Enterprise Services.

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.

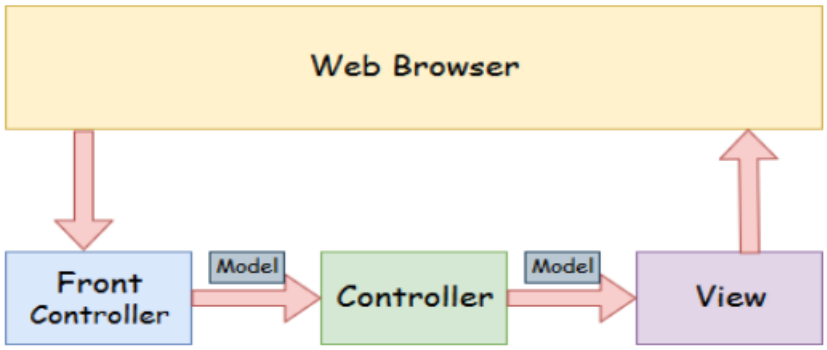
Spring is a lightweight framework. It provides the best mechanisms for different frameworks, such as Struts, Hibernate, EJB, JSF, and Tapestry. It helps solve realtime technical problems. Spring contains multiple modules, such as WEB MVC, IOC, DAO, AOP, Context, and ORM.

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 Springbased projects, such as:
Spring Boot
Spring Cloud
Spring GraphQL

7. Explain the Spring Web MVC Framework design.

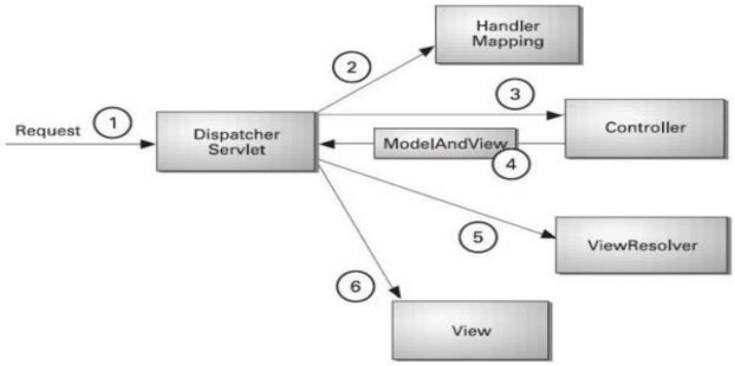
A Spring MVC is a Java framework which is used to build web applications. It follows the Model-View-Controller design pattern. It implements all the basic

features of a core spring framework like Inversion of Control, Dependency Injection. A Spring MVC provides an elegant solution to use MVC in spring framework by the help of DispatcherServlet. Here, DispatcherServlet is a class that receives the incoming request and maps it to the right resource such as controllers, models, and views.



- **Model:** A model contains the data of the application. A data can be a single object or a collection of objects.
- **Controller**: A controller contains the business logic of an application. Here, the @Controller annotation is used to mark the class as the controller.
- **View:** A view represents the provided information in a particular format. Generally, JSP+JSTL is used to create a view page. Although spring also supports other view technologies such as Apache Velocity, Thymeleaf and FreeMarker.
- **Front Controller:** In Spring Web MVC, the DispatcherServlet class works as the front controller. It is responsible to manage the flow of the Spring MVC application.

Understanding the flow of Spring Web MVC



- As displayed in the figure, all the incoming request is intercepted by the DispatcherServlet that works as the front controller.
- The DispatcherServlet gets an entry of handler mapping from the XML file and forwards the request to the controller.
- The controller returns an object of ModelAndView.
- The DispatcherServlet checks the entry of view resolver in the XML file and invokes the specified view component.

8. **What are the benefits of using Spring Security Application?**

Spring Security provides ways to perform authentication and authorization in a web application. We can use spring security in any servlet-based web application.
Some of the benefits of using Spring Security are:

1) Proven technology, it's better to use this than reinvent the wheel. Security is something where we need to take extra care, otherwise our application will be vulnerable for attackers.
2) Prevents some of the common attacks such as CSRF, session fixation attacks.
3) Easy to integrate in any web application. We don't need to modify web application configurations, spring automatically injects security filters to the web application.
4) Provides support for authentication by different ways - in-memory, DAO, JDBC, LDAP and many more.
5) Provides option to ignore specific URL patterns, good for serving static HTML, image files.
6) Support for groups and roles.

Spring Security is a powerful and highly customizable security framework that provides authentication, authorization, and other security features for Spring-based applications. It is a widely used open-source project that helps developers to secure their web applications by implementing security policies and rules. Spring Security provides a set of APIs and classes that can be used to easily configure and implement various security features in web applications. It also provides integration with other Spring Framework components such as Spring MVC, Spring Boot, and others, making it easier to use and integrate with existing applications.

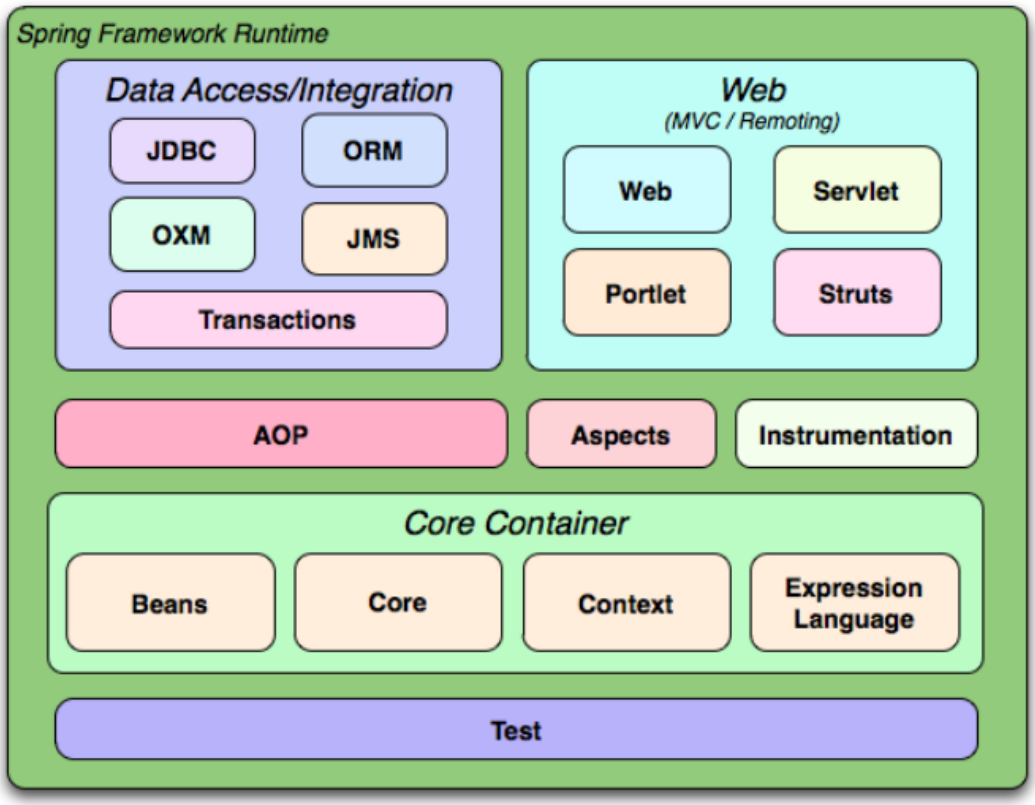
Some of the key features of Spring Security include support for multiple authentication mechanisms, such as form-based authentication, token-based authentication, and others, robust authorization capabilities based on roles and permissions, support for various security protocols such as HTTPS, OAuth, and SAML, and comprehensive logging and auditing capabilities for monitoring security-related events. The core of Spring Security is based on a set of filters that intercept and process incoming HTTP requests, allowing developers to define rules for different types of requests and user roles. Spring Security also provides a range of configuration options that can be used to customize its behaviour and integrate it with other Spring modules.

Part - C

1. **Explain the Spring Framework components.**

The Spring Framework Inversion of Control (IoC) component addresses this concern by providing a formalized means of composing disparate components into a fully working application ready for use. The Spring Framework codifies formalized design patterns as first-class objects that you can integrate into your own application(s). Numerous organizations and institutions use the Spring Framework in this manner to engineer robust, maintainable applications.

The Spring Framework consists of features organized into about 20 modules. These modules are grouped into Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation, and Test, as shown in the following diagram.



1) **Core container**
   This contains the fundamental modules that are the cornerstone of the Spring framework.
- **Core (spring-core)** is the core of the framework that power features such as Inversion of Control and dependency injection.
- **Beans (spring-beans)** provides Beanfactory, which is a sophisticated implementation of the factory pattern.
- **Context (spring-context)** builds on Core and Beans and provides a medium to access defined objects. ApplicationContext interface is the core part of the Context module, and the spring-context-support provides support for third-party interactions such as caching, mailing, and template engines.
- **SpEL (spring-expression)** enables users to use the Spring Expression Language to query and manipulate the object graph at runtime.

2) **Data access/integration**
   This includes the modules that are used to handle data access and transaction processing in an application.
   - **JDBC (spring-jdbc)** provides a JDBC abstraction layer that eliminates the need to separate JDBC coding when dealing with databases.
   - **ORM (spring-orm)** are integration layers for popular object-relational mapping API such as JPA, JDO Hibernate.

- **OXM (spring-oxm)** is the abstraction layer that supports Object/XML mapping implementations like JAXB, XStream.
- **JMS (spring-jms)** is the Java Messaging Service module that creates and consumes messages that directly integrate with the Spring messaging module.
- **Transaction (spring-tx)** offers programmatic and declarative transaction management for classes that include special interfaces and POJOs.

3) **Web**

The Web layer relates to modules that power web-based functions in Spring.

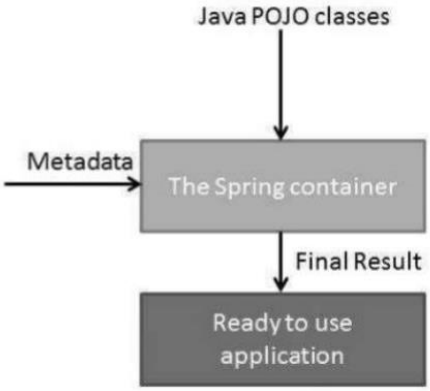
- **WebSocket (spring-websocket)** powers the web socket-based communication for clients and servers.
- **Servlet (spring-webmvc)** is the Spring WebMVC module that contains the MVC and REST implementations.
- **Web (spring-web)** provides all the basic web-oriented features and contains an HTTP client and web-related parts of the Spring remoting.
- **Portlet (spring-webmvc-portlet)** provides the MVC implementation to be used in a portlet environment.

4) **Other Modules**

- **AOP (spring-aop)** provides an aspect-oriented programming implementation that can be used when creating applications.
- **Aspects (spring-aspects)** enables direct integration with the AspectJ programming extension by the eclipse foundation.
- **Instrumentation (spring-instrument)** is the class instrumentation support and class loader implementations for application servers.
- **Messaging (spring-messaging)** provides a robust platform to manage messaging in applications.
- **Test (spring-test)** is the Spring test module that supports unit and integration testing with JUnit and TestNG.

2. **Explain the details of Spring Container type.**

The Spring container is at the core of the Spring Framework. The container will create the objects, wire them together, configure them, and manage their complete life cycle from creation till destruction. The Spring container uses DI to manage the components that make up an application. These objects are called Spring Beans. The container gets its instructions on what objects to instantiate, configure, and assemble by reading the configuration metadata provided. The configuration metadata can be represented either by XML, Java annotations, or Java code. The following diagram represents a high-level view of how Spring works. The Spring IoC container makes use of Java POJO classes and configuration metadata to produce a fully configured and executable system or application.



Spring provides the following two distinct types of containers.

- **Spring BeanFactory Container:**
  This is the simplest container providing the basic support for DI and is defined by the org.springframework.beans.factory.BeanFactory interface. The BeanFactory and related interfaces, such as BeanFactoryAware, InitializingBean, DisposableBean, are still present in Spring for the purpose of backward compatibility with a large number of thirdparty frameworks that integrate with Spring.

- **Spring ApplicationContext Container:**
  This container adds more enterprise-specific functionality such as the ability to resolve textual messages from a properties file and the ability to publish application events to interested event listeners. This container is defined by the org.springframework.context.ApplicationContext interface.

The ApplicationContext container includes all functionality of the BeanFactorycontainer, so it is generally recommended over BeanFactory. BeanFactory can still be used for lightweight applications like mobile devices or applet-based applications where data volume and speed are significant.