**JAVA ACCESSMENT -04**

**NAME:VINAY REDDY SABBELLA**

**THEORITICAL QUESTIONS AND ANSWERS:**

**Q1.What are the major features in different versions of Spring Framework?**

Ans:

1.Version Spring 2.5

2.Spring 3.0

3.Spring 4.0

Feature This version was released in 2007. It was the first version which supported annotations. This version was released in 2009. It made fullfledged use of improvements in Java5 and also provided support to JEE6. This version was released in 2013. This was the first version to provide full support to Java 8.

**Q2. What is a Spring Framework.?**

Ans:-

Spring is a powerful open source, application framework created to reduce the complexity of enterprise application development. It is light-weighted and loosely coupled. It has layered architecture, which allows you to select the components to use, while also providing a cohesive framework for J2EE application development. Spring framework is also called the framework of frameworks as it provides support to various other frameworks such as Struts, Hibernate, EJB, JSF etc.

**Q3. List the advantages of Spring Framework.?**

Ans:-

1.Spring is non invasive: That means you no need to implements any interface or inherit any class from spring to your classes, so when ever you want to change from spring to any other technology then you no need to change the logics of your class.

2.Spring is light weight: Spring is vast framework so spring people divide the whole spring in to different modules, they are designed in such a way that no module is dependent to other module , excep Spring core module, so according to your requirement you can learn a particular module, you no need to learn whole total framework.

3.End to end Development : Spring supports all aspects of application development, Business aspects, persistence aspects, etc, so we can develop a complete application using spring.

4.Spring supports All types of application development: We can develop any type of applications using spring, eg: Core java, web Application, Distributed application, Enterprise application.

5.Spring is versatile: We can integrate any technologies with spring , so we can say spring is versatile,

6.Spring supports dependency injection: The dependency between classes are managed by spring .

**Q4. What are the different features of Spring Framework?**

Ans:-

Following are some of the major features of Spring Framework :

1.Lightweight: Spring is lightweight when it comes to size and transparency. Inversion of control (IOC): The objects give their dependencies instead of creating or looking for dependent objects. This is called Inversion Of Control.

2.Aspect oriented Programming (AOP): Aspect oriented programming in Spring supports cohesive development by separating application business logic from system services.

3.Container: Spring Framework creates and manages the life cycle and configuration of the application objects. MVC Framework: Spring Framework’s MVC web application framework is highly configurable. Other frameworks can also be used easily instead of Spring MVC Framework.

4.Transaction Management: Generic abstraction layer for transaction management is provided by the Spring Framework.

5.Spring’s transaction support can be also used in container less environments.

6.JDBC Exception Handling: The JDBC abstraction layer of the Spring offers an exception hierarchy, which simplifies the error handling strategy.

**Q5. How many modules are there in Spring Framework and what are they?**

Ans

There are modules which are generalized into Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation and Test.

1.Spring Core Container – This layer is basically the core of Spring Framework. It contains the following modules :

Spring Core Spring Bean SpEL (Spring Expression Language) Spring Context

2.Data Access/Integration – This layer provides support to interact with the database.

It contains the following modules :

1.JDBC (Java DataBase Connectivity)

2.ORM (Object Relational Mapping)

3.OXM (Object XML Mappers)

4.JMS (Java Messaging Service) Transactions.

3.Web – This layer provides support to create web application. It contains the following modules

4.Web Web – MVC Web – Socket Web – Portlet.

Aspect Oriented Programming (AOP) – In this layer you can use Advices, Pointcuts etc., to decouple the code. Instrumentation – This layer provides support to class instrumentation and classloader implementations.

Test – This layer provides support to testing with JUnit and TestNG.

Few Miscellaneous modules are given below:

1.Messaging – This module provides support for STOMP. It also supports an annotation programming model that is used for routing and processing STOMP messages from WebSocket clients.

2.Aspects – This module provides support to integration with AspectJ.

**Q6.What is a Spring configuration file?**

Ans:-A Spring configuration file is an XML file. This file mainly contains the classes information. It describes how those classes are configured as well as introduced to each other. The XML configuration files, however, are verbose and more clean. If it’s not planned and written correctly, it becomes very difficult to manage in big projects.

**Q7. What are the different components of a Spring application?**

Ans:-

A Spring application, generally consists of following components:

1.Interface: It defines the functions.

2.Bean class: It contains properties, its setter and getter methods, functions etc.

3.Spring Aspect Oriented Programming (AOP): Provides the functionality of cross-cutting concerns.

4.Bean Configuration File: Contains the information of classes and how to configure then.

5.User program: It uses the function.

**Q8.What are the various ways of using Spring Framework?**

Spring Framework can be used in various ways. They are listed as follows:

1.As a Full-fledged Spring web application. 2.Asathirdpartywebframework,usingSpringFrameworksmiddle-tier.

3.For remote usage. 4.AsEnterpriseJavaBeanwhichcuanwrapexistingPOJOs(PlainOldJava Objects).

**Q9. What is Spring IOC Container?**

Ans:-

At the core of the Spring Framework, lies the Spring container.

The container creates the object, wires them together, configures them and manages their complete life cycle.

The Spring container makes use of Dependency Injection to manage the components that make up an application.

The container receives instructions for which objects to instantiate, configure, and assemble by reading the configuration metadata provided. This metadata can be provided either by XML, Java annotations or Java code.

Q10. What do you mean by Dependency Injection?

Ans:-

In Dependency Injection, you do not have to create your objects but have to describe how they should be created. You don’t connect your components and services together in the code directly, but describe which services are needed by which components in the configuration file. The IoC container will wire them up together.

**PRACTICAL QUESTIONS AND ANSWERS:**

**Implement spring actuator to monitor one microservice?**

ANSWER:

<HTTPS://localhost:8080/actuator> is the url and I got

{"\_links":{"self":{"href":"http://localhost:8080/actuator","templated":false},"health":{"href":"http://localhost:8080/actuator/health","templated":false},"health-path":{"href":"http://localhost:8080/actuator/health/{\*path}","templated":true}}}

As the output.

**Write a code for YML configuration with jpa configuration?**

Application.yml

spring:

datasource:

driverClassName: com.mysql.jdbc.Driver

url: jdbc:mysql://localhost:3306/sampledb7

username: root

password: Admin@123

jpa:

hibernate.ddl-auto: update

generate-ddl: true

show-sql: true

and I wrote one entity class and configured with jpa named as student.class

**Please create and deploy one microservice using Spring CLI?**

->

Step 1: Check Spring boot version

spring version

Spring CLI v2.6.2

Step 2: hello.groovy

This is my Controller

@RestController

class WebApplication {

@RequestMapping("/")

String home() {

"Hello World!"

}

}

Step3: To compile and run the application, type the following command:

spring run hello.groovy.

To set JVM command line arguments, you can use the JAVA\_OPTS environment variable, as shown

JAVA\_OPTS=-Xmx1024m spring run hello.groovy

**How to change default server port from 8080 to 9090?**

ANSWER:

**Using Property Files**

The fastest and easiest way to customize Spring Boot is by overriding the values of the default properties.

For the server port, the property we want to change is server.port.

By default, the embedded server starts on port 8080.

So, let's see how to provide a different value in an application.properties file:

server.port=8081

Now the server will start on port 8081.

And we can do the same if we're using an application.yml file:

server:

port : 8081

Both files are loaded automatically by Spring Boot if placed in the src/main/resources directory of a Maven application.

**Environment Specific Ports**

If we have an application deployed in different environments, we may want it to run on different ports on each system.

We can easily achieve this by combining the property files approach with Spring profiles. Specifically, we can create a property file for each environment.

For example, we'll have an application-dev.properties file with this content:

server.port=8081

Then we'll add another application-qa.properties file with a different port:

server.port=8082

Now, the property files configuration should be sufficient for most cases. However, there are other options for this goal, so let's explore them as well.

**Programmatic Configuration**

We can configure the port programmatically either by setting the specific property when starting the application or by customizing the embedded server configuration.

First, let's see how to set the property in the main @SpringBootApplication class:

@SpringBootApplication

public class CustomApplication {

public static void main(String[] args) {

SpringApplication app = new SpringApplication(CustomApplication.class);

app.setDefaultProperties(Collections

.singletonMap("server.port", "8083"));

app.run(args);

}

}

Next, to customize the server configuration, we have to implement the WebServerFactoryCustomizer interface:

@Component

public class ServerPortCustomizer

implements WebServerFactoryCustomizer<ConfigurableWebServerFactory> {

@Override

public void customize(ConfigurableWebServerFactory factory) {

factory.setPort(8086);

}

}

Note that this applies to the Spring Boot 2.x version.

For Spring Boot 1.x, we can similarly implement the EmbeddedServletContainerCustomizer interface.

**Using Command-Line Arguments**

When packaging and running our application as a jar, we can set the server.port argument with the java command:

java -jar spring-5.jar --server.port=8083

or by using the equivalent syntax:

java -jar -Dserver.port=8083 spring-5.jar

**How to resolve white label error page in spring boot application?**

**Answer:**

First, let's see how we can disable the white label error page entirely,

by setting the server.error.whitelabel.enabled property to false:

server.error.whitelabel.enabled=false

Adding this entry to the application.properties file will disable the error page

and show a concise page that originates from the underlying application container, e.g., Tomcat.

We can achieve the same result by excluding the ErrorMvcAutoConfiguration bean.

We can do this by either adding this entry to the properties file:

spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.web.ErrorMvcAutoConfiguration

for Spring Boot 2.0

#pring.autoconfigure.exclude=org.springframework.boot.autoconfigure.web.servlet.error.ErrorMvcAutoConfiguration

Or by adding this annotation to the main class:

@EnableAutoConfiguration(exclude = {ErrorMvcAutoConfiguration.class})

All the methods mentioned above will disable the white label error page.