HW#2 Gulnisa Aslan….gulnisaslan@gmail.com

1 – IOC and DI means ?

IoC is a generic term meanin rather than having the application call the methods in a framework, the framework calls implementations provided by the application.

DI is a from of IoC, where implementations are passed into an object through constructors/setter/service look ups, which the object will depend on inn order to behave correctly.

---

2 – Spring Bean Scopes ?

When defining a <bean> you have the option of declaring a scope for that bean. For example, to force Spring to produce a new bean instance each time one is needed, you should declare the bean's scope attribute to be **prototype**. Similarly, if you want Spring to return the same bean instance each time one is needed, you should declare the bean's scope attribute to be **singleton**.

The Spring Framework supports the following five scopes, three of which are available only if you use a web-aware ApplicationContext.

Sr.No.	Scope & Description
1	**singleton** This scopes the bean definition to a single instance per Spring IoC container (default).
2	**prototype** This scopes a single bean definition to have any number of object instances.

3	**request** This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext.
4	**session** This scopes a bean definition to an HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.
5	**global-session** This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.

---

3 – What does @SpringBootApplication do ?

Many Spring Boot developers like their apps to use auto-configuration, component scan and be able to define extra configuration on their "application class". A single @SpringBootApplication annotation can be used to enable those three features, that is:

- @EnableAutoConfiguration: enable Spring Boot's auto-configuration mechanism
- @ComponentScan: enable @Component scan on the package where the application is located (see the best practices)
- @Configuration: allow to register extra beans in the context or import additional configuration classes
- **********

4 – What is Spring AOP ? Where and How to use it ?

Spring AOP enables Aspect-Oriented Programming in spring applications. In AOP, aspects enable
the modularization of concerns such as transaction management, logging or security that cut across multiple types and objects (often termed **crosscutting concerns**).

AOP provides the way to dynamically add the cross-cutting concern before, after or around the actual logic
using simple pluggable configurations. It makes easy to maintain code in the present and future as well. You can add/remove concerns without recompiling complete sourcecode simply by changing configuration files (if you are applying aspects suing XML configuration).

---

5 – What is Singleton and where to use it ?

Swagger2 is an open source project used to generate the REST API documents for RESTful web services. It provides a user interface to access our RESTful web services via the web browser.

To enable the Swagger2 in Spring Boot application, you need to add the following dependencies in our build configurations file.

****

6 – What is Spring Boot Actuator and Where to use it ?

Spring Boot Actuator is a sub-project of the Spring Boot Framework. It includes a number of additional features that help us to monitor and manage the Spring Boot application. It contains the actuator endpoints (the place where the resources live). We can
use **HTTP** and **JMX** endpoints to manage and monitor the Spring Boot application. If we want to get production-ready features in an application, we should use the S**pring Boot actuator.**

Spring Boot Actuator Features

There are **three** main features of Spring Boot Actuator:

- **Endpoints**
- **Metrics**
- **Audit**

Endpoint: The actuator endpoints allows us to monitor and interact with the application. Spring Boot provides a number of built-in endpoints. We can also create our own endpoint. We can enable and disable each endpoint individually. Most of the application choose **HTTP**, where the Id of the endpoint, along with the prefix of **/actuator,** is mapped to a URL.

For example, the **/health** endpoint provides the basic health information of an application. The actuator, by default, mapped it to **/actuator/health**.

Metrics: Spring Boot Actuator provides dimensional metrics by integrating with the **micrometer**. The micrometer is integrated into Spring Boot. It is the instrumentation library powering the delivery of application metrics from Spring. It provides vendor-neutral interfaces for **timers, gauges, counters, distribution summaries,** and **long task timers** with a dimensional data model.

Audit: Spring Boot provides a flexible audit framework that publishes events to an **AuditEventRepository.** It automatically publishes the authentication events if spring-security is in execution.

****

7 - What is the primary difference between Spring and Spring Boot ?

S.No.	Spring	Spring Boot
1.	Spring is an open-source lightweight framework widely used to develop enterprise applications.	Spring Boot is built on top of the conventional spring framework, widely used to develop REST APIs.
2.	The most important feature of the Spring Framework is dependency injection.	The most important feature of the Spring Boot is Autoconfiguration.
3.	It helps to create a loosely coupled application.	It helps to create a stand-alone application.
4.	To run the Spring application, we need to set the server explicitly.	Spring Boot provides embedded servers such as Tomcat and Jetty etc.
5.	To run the Spring application, a deployment descriptor is required.	There is no requirement for a deployment descriptor.
6.	To create a Spring application, the developers write lots of code.	It reduces the lines of code.
7.	It doesn’t provide support for the in-memory database.	It provides support for the in-memory database such as H2.

****

8 – Why to use VCS ?

In software enginnering, **version control** (also known as **revision control**, **source control**, or **source code management**) is a class of systems responsible for managing changes to computer programs, documents, large web sites, or other collections of information. Version control is a component of software configuration management.[1]

Changes are usually identified by a number or letter code, termed the "revision number", "revision level", or simply "revision". For example, an initial set of files is "revision 1". When the first change is made, the resulting set is "revision 2", and so on. Each revision is associated with a timestamp and the person making the change. Revisions can be compared, restored, and with some types of files, merged.

The need for a logical way to organize and control revisions has existed for almost as long as writing has existed, but revision control became much more important, and complicated, when the era of computing began. The numbering of book editions and of specification revisions are examples that date back to the print-only era. Today, the most capable (as well as complex) revision control systems are those used in software development, where a team of people may concurrently make changes to the same files.

---

9 – What are SOLID Principles ? Give sample usages in Java ?

SOLID principles are class-level, object-oriented design concepts that, in conjunction with an extensive test suite, help you avoid code rot.

SOLID design is an acronym for the following five principles:

- Single Responsibility Principle

The Single Responsibility Principle (SRP) states that there should never be more than one reason for a class to change. This means that every class, or similar structure, in your code should have only one job to do.

- Open-Closed Principle

The Open-Closed Principle (OCP) states that classes should be open for extension but closed for modification. “Open to extension” means that you should design your classes so that new functionality can be added as new requirements are generated. “Closed for modification” means that once you have developed a class you should never modify it, except to correct bugs.

Liskov Substitution Principle

The Liskov Substitution Principle (LSP) applies to inheritance hierarchies, specifying that you should design your classes so that client dependencies can be substituted with subclasses without the client knowing about the change.

-

- Interface Segregation Principle

The Interface Segregation Principle (ISP) states that clients should not be forced to depend upon interface members they do not use. When we have non-cohesive interfaces, the ISP guides us to create multiple, smaller, cohesive interfaces.

- Dependency Inversion Principle

The Dependency Inversion Principle (DIP) states that high-level modules should not depend upon low-level modules; they should depend on abstractions.

Secondly, abstractions should not depend upon details; details should depend upon abstractions. The idea is that we isolate our class behind a boundary formed by the abstractions it depends on. If all the The Dependency Inversion Principle (DIP) states that high-level modules should not depend upon low-level modules; they should depend on abstractions.

Secondly, abstractions should not depend upon details; details should depend upon abstractions. The idea is that we isolate our class behind a boundary formed by the abstractions it depends on. If all the details behind those abstractions change, then our class is still safe. This helps keep coupling low and makes our design easier to change. DIP also allows us to test things in isolation.

details behind those abstractions change, then our class is still safe. This helps keep coupling low and makes our design easier to change. DIP also allows us to test things in isolation.

---

10 - What is RAD model ?

In software engineering, **version control** (also known as **revision control**, **source control**, or **source code management**) is a class of systems responsible for managing changes to computer programs, documents, large web sites, or other collections of information. Version control is a component of software configuration management.[1]

Changes are usually identified by a number or letter code, termed the "revision number", "revision level", or simply "revision". For example, an initial set of files is "revision 1". When the first change is made, the resulting set is "revision 2", and so on. Each revision is associated with a timestamp and the person making the change. Revisions can be compared, restored, and with some types of files, merged.

The need for a logical way to organize and control revisions has existed for almost as long as writing has existed, but revision control became much more important, and complicated, when the era of computing began. The numbering of book editions and of specification revisions are examples that date back to the print-only era. Today, the most capable (as well as complex) revision control systems are those used in software development, where a team of people may concurrently make changes to the same files.

---

11 - What is Spring Boot starter ? How is it useful ?

Spring Boot provides a number of **starters** that allow us to add jars in the classpath. Spring Boot built-in **starters** make development easier and rapid. **Spring Boot Starters** are the **dependency descriptors**.

In the Spring Boot Framework, all the starters follow a similar naming pattern: **spring-boot-starter-***, where * denotes a particular type of application. For example, if we want to use Spring and JPA for database access, we need to include the **spring-boot-starter-data-jpa** dependency in our **pom.xml** file of the project.

---

12 – What is Caching ? How can we achive caching in Spring Boot ?

Caching is a mechanism to enhance the performance of a system. It is a temporary memory that lies between the application and the persistent database. Cache memory stores recently used data items in order to reduce the number of database hits as much as possible.

We use a cache to protect the database or to avoid cost-intensive calculations. Spring provides an abstraction layer for implementing a cache.

1. Navigate to https://start.spring.io. This service pulls in all the dependencies you need for an application and does most of the setup for you.
2. Choose either Gradle or Maven and the language you want to use. This guide assumes that you chose Java.
3. Click **Dependencies** and select **Spring cache abstraction**.
4. Click **Generate**.

---

13 – What & How & Where & Why to logging ?

Logging in application runtime is storage method to can readable,controlable statu of systematic.

We can use debugging and testing processses in logging development.

Logging, to be systematic and can be controlable.Logging, to explain in status of application.

---

14 - What is Swagger? Have you implemented it using Spring Boot?

Swagger2 is an open source project used to generate the REST API documents for RESTful web services. It provides a user interface to access our RESTful web services via the web browser.

To enable the Swagger2 in Spring Boot application, you need to add the following dependencies in our build configurations file.

Using Swagger:

We add two dependencies pom.xml file

```
<dependency>
   <groupId>io.springfox</groupId>
   <artifactId>springfox-swagger2</artifactId>
   <version>2.7.0</version>
</dependency>
<dependency>
   <groupId>io.springfox</groupId>
   <artifactId>springfox-swagger-ui</artifactId>
   <version>2.7.0</version>
</dependency>
```

```
Last we add  @EnableSwagger annotation and
@Bean
   public Docket productApi() {
```

```
      return new Docket(DocumentationType.SWAGGER_2).select()

.apis(RequestHandlerSelectors.basePackage("com.tutorialspoint.swa
ggerdemo")).build(); code ……..Application.
```

```
package com.tutorialspoint.swaggerdemo;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import
springfox.documentation.swagger2.annotations.EnableSwagger2;

@SpringBootApplication
@EnableSwagger2
public class SwaggerDemoApplication {
   public static void main(String[] args) {
      SpringApplication.run(SwaggerDemoApplication.class, args);
   }
   @Bean
   public Docket productApi() {
      return new Docket(DocumentationType.SWAGGER_2).select()

.apis(RequestHandlerSelectors.basePackage("com.tutorialspoint.swa
ggerdemo")).build();
```