**Q1. what is the @component and @Controller**.

Ans->

@Component: It is a class-level annotation. It is used to mark a Java class as a bean.

A Java class annotated with @Component is found during the classpath.

The Spring Framework pick it up and configure it in the application context as a Spring Bean.

@Controller: The @Controller is a class-level annotation.

It is a specialization of @Component.

It marks a class as a web request handler.

It is often used to serve web pages.

By default, it returns a string that indicates which route to redirect.

It is mostly used with @RequestMapping annotation.

**Q2. @RequestMapping,@Required,@Qualifier,@Autowired @Temporal,@Entity @RequestBody @RestController @Query @PathVariable Annotation, explain?**

Ans->

@RequestMapping:

It is used to map the web requests.

It has many optional elements like consumes, header, method, name, params, path, produces, and value.

We use it with the class as well as the method.

@Required:

It applies to the bean setter method.

It indicates that the annotated bean must be populated at configuration time with the required property,

else it throws an exception BeanInitilizationException.

@Qualifier:

annotation is used to resolve the autowiring conflict, when there are multiple beans of same type.

The @Qualifier annotation can be used on any class annotated with @Component or on methods annotated with @Bean .

This annotation can also be applied on constructor arguments or method parameters.

@Autowired: Spring provides annotation-based auto-wiring by providing @Autowired annotation.

It is used to autowire spring bean on setter methods, instance variable, and constructor.

When we use @Autowired annotation, the spring container auto-wires the bean by matching data-type

@Temporal:

annotation to insert date, time or both in database table.

Using TemporalType we can insert data, time or both int table.

@Entity:

annotation specifies that the class is an entity and is mapped to a database table. The

@Table annotation specifies the name of the database table to be used for mapping.

@RequestBody:

It is used to bind HTTP request with an object in a method parameter.

Internally it uses HTTP MessageConverters to convert the body of the request.

When we annotate a method parameter with @RequestBody, the Spring framework binds

the incoming HTTP request body to that parameter.

@PathVariable:

It is used to extract the values from the URI.

It is most suitable for the RESTful web service, where the URL contains a path variable.

We can define multiple @PathVariable in a method.

@RestController:

It can be considered as a combination of @Controller and @ResponseBody annotations.

The @RestController annotation is itself annotated with the @ResponseBody annotation.

It eliminates the need for annotating each method with @ResponseBody.

@Quer:

In order to define SQL to execute for a Spring Data repository method, we can annotate the method with the

@Query annotation its value attribute contains the JPQL or SQL to execute.

The @Query annotation takes precedence over named queries

which are annotated with @NamedQuery or defined in an orm

**Q3. What are the Stereotype annotations and define MVC flow.**

Ans->

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.

Spring Framework provides us with some special annotations.

These annotations are used to create Spring beans automatically in the application context.

@Component annotation is the main Stereotype Annotation.

There are some Stereotype meta-annotations which is derived from @Component those are

@Service:

We specify a class with @Service to indicate that they’re holding the business logic.

Besides being used in the service layer, there isn’t any other special use for this annotation.

The utility classes can be marked as Service classes.

@Repository:

We specify a class with @Repository to indicate that they’re dealing with CRUD operations, usually,

it’s used with DAO (Data Access Object) or Repository implementations that deal with database tables.

@Controller:

We specify a class with @Controller to indicate that they’re front controllers and responsible

to handle user requests and return the appropriate response. It is mostly used with REST Web Services.

**Q4.- Difference between application.properties & YML file.**

Ans->

.properties stores data in sequential format, whereas

.yml stores data in hierarchical format.

.properties supports only key-value pairs (basically string values), whereas

.yml supports key-value pair, as well as map, list & scalar type values.

.properties is specifically used by Java, whereas

.yml can be used by other languages (eg Java, Python, ROR, etc).

**Q5. What are the scopes in Spring Framework?**

Ans-> When defining a <bean> we 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, we should declare the bean's scope attribute to be prototype. Similarly, if we want Spring to return the same bean instance each time one is needed, we should declare the bean's scope attribute to be singleton.

The Spring Framework supports the following five scopes.

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.

**Q6. What is cascading and what are different types of cascading?**

Ans->

To establish a dependency between related entities, JPA provides javax.persistence.CascadeType enumerated types that define the cascade operations.

These cascading operations can be defined with any type of mapping

i.e. One-to-One, One-to-Many, Many-to-One, Many-to-Many.

Cascade Operation:

ALL:

In this case, all the above cascade operations can be applied to the entities related to parent entity.

PERSIST:

In this cascade operation, if the parent entity is persisted then all its related entity will also be persisted.

MERGE:

In this cascade operation, if the parent entity is merged then all its related entity will also be merged.

REMOVE:

In this cascade operation, if the parent entity is removed then all its related entity will also be removed.

REFRESH:

In this cascade operation, if the parent entity is refreshed then all its related entity will also be refreshed.

DETACH:

In this cascade operation, if the parent entity is detached then all its related entity will also be detached.

**Q7.what is the ioc container and autowiring?**

Ans-> An IoC container is a common characteristic of frameworks that implement IoC. In the Spring framework, the interface ApplicationContext represents the IoC container. The Spring container is responsible for instantiating, configuring and assembling objects known as beans, as well as managing their life cycles.

The @Autowired annotation provides more fine-grained control over where and how autowiring should be accomplished. The @Autowired annotation can be used to autowire bean on the setter method just like @Required annotation, constructor, a property or methods with arbitrary names or multiple arguments.

**Q8.What is difference between hibernate and jpa? and What is Hibernate cache?.**

Ans->

A JPA (Java Persistence API) is a specification of Java which is used to access, manage, and persist data

between Java object and relational database. It is considered as a standard approach for Object Relational Mapping.

JPA can be seen as a bridge between object-oriented domain models and relational database systems.

Being a specification, JPA doesn't perform any operation by itself.

Thus, it requires implementation.

So, ORM tools like Hibernate, TopLink, and iBatis implements JPA specifications for data persistence.

A Hibernate is a Java framework which is used to store the Java objects in the relational database system.

It is an open-source, lightweight, ORM (Object Relational Mapping) tool.

Hibernate is an implementation of JPA. So, it follows the common standards provided by the JPA.

**Q9.what is difference between Named Queries and Criteria Queries?**

Ans->

Named query is the way you define your query by giving it a name. You could

define this in mapping file in hibernate or also using annotations at entity level.

Criteria query refers to actual sql queries (referring to actual database objects).

These queries are the sql statements which can be directly executed in database using a database client.

Q1. Please write a program to consume Employee record from service-1 to service-3   
Q2. Please Develop a restful webservice to perform CRUD operations. Entities should have Student , Courses and Teachers.  
Q3 Develop CRUD (Student, Course, Teachers entities) from csv to insert into main tables.   
            Development should cover below  
            YAML Configuration  
            Spring JPA  
            SpringData  
            Unit Testing                       
            Lombook Or Sleuth  
            Acutator(Metrics & health check URL configuration)  
Q4. deploy the above application in Linux Machine using maven or gradle?  
Q5.  How Rest API Call/consume happens in SpringBoot? RestTemplate or Feign Client  
Q6.What is Junit and annotations used in junit? write all test cases for above given example

Q7.List Annotations from springboot?

Ans->

@EnableAutoConfiguration: It auto-configures the bean that is present in the classpath and configures it to run the methods.

The use of this annotation is reduced in Spring Boot 1.2.0 release because developers provided an alternative of the annotation, i.e. @SpringBootApplication.

@SpringBootApplication: It is a combination of three annotations @EnableAutoConfiguration, @ComponentScan, and @Configuration.