WEEKLY ACCESSMENT 06

NAME:VINAY REDDY SABBELLA

THEORITICAL QUESTIONS:

Q1. what is the @component and @Controller ?

Answer:

@Component is an annotation that allows Spring to automatically detect our custom beans. In other words, without having to write any explicit code, Spring will: Scan our application for classes annotated with @Component. Instantiate them and inject any specified dependencies into them.

@Controller annotation indicates that a particular class serves the role of a controller. There is no need to extend any controller base class or reference the Servlet API. The basic purpose of the @Controller annotation is to act as a stereotype for the annotated class, indicating its role. The dispatcher will scan such annotated classes for mapped methods, detecting @RequestMapping annotations

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

Answer:

**@RequestMapping** is one of the most widely used **Spring MVC** annotation. org.springframework.web.vind.annotation.RequestMaping annotation is used to map web requests onto specific handler classes and/or handler methods.

**@RequestMapping with @PathVariable**: RequestMapping annotation can be used to handle dynamic URIs where one or more of the URI value works as a parameter. We can even specify [Regular Expression](https://www.journaldev.com/634/regular-expression-in-java-regex-example) for URI dynamic parameter to accept only specific type of input. It works with **@PathVariable annotation** through which we can map the URI variable to one of the method arguments.

By using the @Qualifier annotation, we can **eliminate the issue of which bean needs to be injected**. By including the @Qualifier annotation, together with the name of the specific implementation we want to use.

@Required annotation applies to bean property setter methods and it indicates that the affected bean property must be populated in XML configuration file at configuration time. Otherwise, the container throws a BeanInitializationException exception.

@Autowired annotation is optional for constructor based injection. Here, the person object from the container is passed to the constructor while creating the Customer object. The setter method will be called with the Person object at runtime by the container.

@Temporal annotation has the single parameter value of type TemporalType. It can be either DATE, TIME or TIMESTAMP, depending on the underlying SQL type that we want to use for the mapping.

@RequestBody**annotation maps the**HttpRequest**body to a transfer or domain object, enabling automatic deserialization** of the inbound HttpRequest body onto a Java object.

@Entity annotation says that table which has to be created in the database when we write a entity class in the project.

@RestController is a specialized version of the controller. It includes the @Controller and @ResponseBody annotations, and as a result, simplifies the controller implementation.

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.xml file.

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

Answer:

Stereotype annotations are markers for any class that fulfills a role within an application. This helps remove, or at least greatly reduce, the Spring XML configuration required for these components.

The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. ... MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.

Q4.- Diffrence between application.properties & YML fiile.

Answer:

1. Content format comparison:  
.properties file, pass.Come to the connection; pass=To assign a value, the structure, there is no hierarchical feel, but it is more directly.  
.yml file, pass：To lay lay, the structure, there is a more obvious level of layering, and finally the key is assigned: after need to leave a space

2. Execute the order  
If the application.properties file and Application.yml files are present, the YML file will be loaded first, then the Loaded Properties file overrides the YML file. So in the recommended project, only one of the types of files can be used.

Q5. What are the scopes in Spring Framework?

Answer:

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. |

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

Answer:

**Cascading** is a phenomenon involving one object propagating to other objects via a relationship. It is transitive in nature and the cascade attribute in hibernate defines the relationship between the entities. The cascading types supported by the hibernate framework are as follow:

* CascadeType.PERSIST: It means that the save() and persist() operations in the hibernate cascade to the related entities
* CascadeType.MERGE: It means that the related entities are joined when the owning entity is joined
* CascadeType.REMOVE: It means that the related entities are deleted when the owning entity is deleted
* CascadeType.DETACH: It detaches all the related entities if a *manual detach* occurs
* CascadeType.REFRESH: It works similar to the refresh() operation in the hibernate
* CascadeType.ALL: It is an alternative for performing all the above cascade operations in the hibernate framework

Q7.what is the ioc container and autowiring ?

Answer:

The IoC container is responsible to instantiate, configure and assemble the objects. The IoC container gets informations from the XML file and works accordingly. The main tasks performed by IoC container are:

* to instantiate the application class
* to configure the object
* to assemble the dependencies between the objects

Autowiring feature of spring framework **enables you to inject the object dependency implicitly**. It internally uses setter or constructor injection. Autowiring can't be used to inject primitive and string values. It works with reference only.

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

Answer:

JPA vs. Hibernate

|  |  |
| --- | --- |
| **JPA** | **Hibernate** |
| Java Persistence API (JPA) defines the management of relational data in the Java applications. | Hibernate is an Object-Relational Mapping (ORM) tool which is used to save the state of Java object into the database. |
| It is just a specification. Various ORM tools implement it for data persistence. | It is one of the most frequently used JPA implementation. |
| It is defined in **javax.persistence** package. | It is defined in **org.hibernate** package. |
| The **EntityManagerFactory** interface is used to interact with the entity manager factory for the persistence unit. Thus, it provides an entity manager. | It uses **SessionFactory** interface to create Session instances. |
| It uses **EntityManager** interface to create, read, and delete operations for instances of mapped entity classes. This interface interacts with the persistence context. | It uses **Session** interface to create, read, and delete operations for instances of mapped entity classes. It behaves as a runtime interface between a Java application and Hibernate. |
| It uses **Java Persistence Query Language** (JPQL) as an object-oriented query language to perform database operations. | It uses **Hibernate Query Language** (HQL) as an object-oriented query language to perform database operations. |

Hibernate Cache:

Hibernate caching improves the performance of the application by pooling the object in the cache. It is useful when we have to fetch the same data multiple times.

There are mainly two types of caching:

* First Level Cache, and
* Second Level Cache

#### **First Level Cache**

Session object holds the first level cache data. It is enabled by default. The first level cache data will not be available to entire application. An application can use many session object.

#### **Second Level Cache**

SessionFactory object holds the second level cache data. The data stored in the second level cache will be available to entire application. But we need to enable it explicitely.

Q10.what is difference between Named Queries and Criteria Queries?

Answer:

HQL is **suitable for executing Static Queries**, where as Criteria is suitable for executing Dynamic Queries. HQL is to perform both select and non-select operations on the data, Criteria is only for selecting the data, we cannot perform non-select operations using criteria.