Introduction to spring data JPA: refer
Introduction to microservices: refer
Hibernate in depth: refer
important method(s)/Classes in the Hibernate which are imp from interview perspective
- *Methods* of EntityManager:flush(),detach(Object),clear(),refresh()
- *Annotations* @Entity,@Id,@GeneratedValue,@Column(nullable =false),@CreationTimestamp, @UpdateTimestamp,@OnetoOne,@OneToMay,@ManyToMany,@ManyToOne(fetch=FetchType.LAZY), @NamedQuery(name="find_all_courses", query= "select c from Course c"),@Transactional,@PersistenceContext

- Any relation ending with __**ToOne** (ManyToOne,OneToOne) is

always **EAGER** FETCHING and Relation ending with **__ToMany** (ManyToMany,OneToMany) is always **LAZY** FETCHING

JPA inheritance Hierarchy and mapping: refer
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
@Inheritance(strategy = InheritanceType.JOINED)

Hibernate queries(writing JPQL by different formats query, namedQuery, nativeQuery)
- **JPQL**(Java PersistenceQuery Language): queries
JPA Queries(interface of hibernate, JPA provides specification)
- Example: queries

Caching in Hibernate: refer

Imp spring boot snippets:

Creating global exception handlers: refer

Create Interceptors (required for filtering or anything you want to do before the request is actually sent to business layer for processing): refer

AOP (aspect oriented programming)in spring boot:
Creating aspect class: refer
Note: @Before, @After and @Arond are the pointcut expressions that match the joinpoint

SpringBoot Controller: refer
RowMapper(required to read data from db and populate in ab object): refer

On the fly revision questions:
What is the difference between @Controller and @RestController