# Spring Data JPA

1.How to do pagination when you use Spring Data JPA ?  
Ans - Instead of CrudRepository<Product, Integer> use PagingAndSortingRepository<Product, Integer>.

//See the sample code below   
//public interface ProductRepository extends CrudRepository<Product, Integer> {

public interface ProductRepository extends PagingAndSortingRepository<Product, Integer> {

List<Product> findByName(String name);

List<Product> findByNameAndDesc(String name, String desc);

List<Product> findByPriceGreaterThan(double price);

List<Product> findByDescContains(String contains);

List<Product> findByPriceIsBetween(Double price1, Double price2);

List<Product> findByDescLike(String pattern);

Page<Product> findAll(Pageable pageable);

}

2. What are the types / classes available in the pagination ?  
Ans - Interfaces - Page, Pageable , Class - PageRequest

@Test

public void testFindAllPaging() {

//Lets first check how many records are there in DB

long totCount = prodRepository.count();

int no\_of\_recs =1;

long totalPages = totCount / no\_of\_recs;

Sort sort = Sort.by("name");

for(int pageNo =0; pageNo < totalPages ;pageNo++) {

Pageable pageable = PageRequest.of(pageNo, no\_of\_recs);

Page<Product> pageProduct = prodRepository.findAll(pageable);

System.out.println("Page\_No ::: " + pageNo);

pageProduct.forEach(prd -> System.out.println(prd));

}

System.out.println("Total no of records in Product table : " + totCount);

}

3. How will you sort result from the database ?  
Ans - Below is the sample code,  
@Test

public void testFindAllSortingAsc() {

//Example of sorting by multiple properties

Sort sort = Sort.by("name","price");

prodRepository.findAll(sort).forEach(prod -> System.out.println(prod.getName()));

}

@Test

public void testFindAllSortingDesc() {

Sort sort = Sort.by("name").descending();

prodRepository.findAll(sort).forEach(prod -> System.out.println(prod.getName()));

}

4. Inheritance mapping

1. Inheritance exists in OOP languages like Java, however inheritance do not exist in the database, To handle this JPA gives three options,

* SINGLE\_TABLE
* TABLE\_PER\_CLASS
* JOINED

Lets look at the TABLE\_PER\_CLASS strategy. In this use case we have below inheritance hierarchy in this scenario,

* Payment class - This is super class
* CreditCard class - This extends the Payment class
* Check class - This also extends the Payment class

And now we are talking about, how to save all this in the database, when we are following TABLE\_PER\_CLASS strategy.

# Caching

EH Cache - Second level cache provider. It will cache objects at Session factory level.