* Super Class and Subclasses have their own table .
* Subclass table only has attributes of its own and not of super class.
* Polymorphic queries gives poor better performance .
* It allows the null constraints.

|  |
| --- |
| Implementation Details |

* strategy=InheritancyType.JOINED.
* Subclases have a primary key that is a foreign key of super class.

package org.learnhibernatein5days.on.java.model;

import java.time.LocalDate;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Inheritance;

import javax.persistence.InheritanceType;

import javax.persistence.Version;

@Entity

@Inheritance(strategy = InheritanceType.JOINED)

public abstract class Magazine {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

@Column(name = "id", updatable = false, nullable = false)

private Long id;

@Version

private int version;

private String title;

private LocalDate publishedDate;

private int numPages;

public Long getId() {

return this.id;

}

public LocalDate getPublishedDate() {

return publishedDate;

}

public void setPublishedDate(LocalDate publishedDate) {

this.publishedDate = publishedDate;

}

public void setVersion(int version) {

this.version = version;

}

public int getVersion() {

return this.version;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public int getNumPages() {

return numPages;

}

public void setNumPages(int numPages) {

this.numPages = numPages;

}

}

**package** org.learnhibernatein5days.on.java.model;

**import** javax.persistence.Entity;

@Entity

**public** **class** Digital **extends** Magazine {

**private** String url;

**public** String getUrl() {

**return** url;

}

**public** **void** setUrl(String url) {

**this**.url = url;

}

@Override

**public** String toString() {

StringBuilder builder = **new** StringBuilder();

builder.append("Digital []");

**return** builder.toString();

}

}

**package** org.learnhibernatein5days.on.java.model;

**import** javax.persistence.Entity;

@Entity

**public** **class** Printed **extends** Magazine {

**private** **double** weight;

**private** String dimensions;

**public** **double** getWeight() {

**return** weight;

}

**public** **void** setWeight(**double** weight) {

**this**.weight = weight;

}

**public** String getDimensions() {

**return** dimensions;

}

**public** **void** setDimensions(String dimensions) {

**this**.dimensions = dimensions;

}

@Override

**public** String toString() {

StringBuilder builder = **new** StringBuilder();

builder.append("Printed [weight=");

builder.append(weight);

builder.append(", dimensions=");

builder.append(dimensions);

builder.append("]");

**return** builder.toString();

}

}

log.info("... testJoinedTableInheritance ...");

// persist a new Book entity

EntityManager em = emf.createEntityManager();

em.getTransaction().begin();

Digital digital = **new** Digital();

digital.setNumPages(400);

digital.setPublishedDate(LocalDate.*of*(2019, 2, 4));

digital.setTitle("Spring Microservices in action");

digital.setUrl("www.springmicroservices.com");

Printed printed = **new** Printed();

printed.setNumPages(400);

printed.setPublishedDate(LocalDate.*of*(2019, 2, 4));

printed.setTitle("Spring Microservices in action");

printed.setWeight(12.5);

printed.setDimensions("23cmx25cmx2cm");

em.persist(digital);

em.persist(printed);

em.getTransaction().commit();

em.close();

|  |
| --- |
| Query |

log.info("... testJoinedTableInheritance ...");

EntityManager em = emf.createEntityManager();

em.getTransaction().begin();

// read the Digital entity

em = emf.createEntityManager();

em.getTransaction().begin();

TypedQuery<Digital> query = em.createQuery("SELECT d FROM Digital d", Digital.**class**);

List<Digital> digitalList = query.getResultList();

**for**(Digital digital : digitalList)

{

System.***out***.println("Title : " + digital.getTitle() + " Url : " + digital.getUrl());

}

em.getTransaction().commit();

em.close();

|  |
| --- |
| Pointer for Better Performance |

* Polymorphic queries don’t have joins
* Performance is better as join is not involved