Demonstrate lazy collection in Hibernate.

**package** hibernate6;

**import** javax.persistence.\*;

**import** java.util.List;

@Entity

**public** **class** Author {

@Id

@GeneratedValue

**private** Long id;

**private** String name;

@OneToMany(mappedBy = "author", fetch = FetchType.***LAZY***)

**private** List<Book> books;

**public** Author() {

}

**public** Author(String name) {

**this**.name = name;

}

// getters and setters

**public** Long getId() {

**return** id;

}

**public** String getName() {

**return** name;

}

**public** List<Book> getBooks() {

**return** books;

}

}

**package** hibernate6;

**import** javax.persistence.\*;

@Entity

**public** **class** Book {

@Id

@GeneratedValue

**private** Long id;

**private** String title;

@ManyToOne

@JoinColumn(name = "author\_id")

**private** Author author;

**public** Book() {

}

**public** Book(String title, Author author) {

**this**.title = title;

**this**.author = author;

}

// getters and setters

**public** Long getId() {

**return** id;

}

**public** String getTitle() {

**return** title;

}

**public** Author getAuthor() {

**return** author;

}

}

**package** hibernate6;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.cfg.Configuration;

**import** java.util.List;

**public** **class** HibernateLazyCollectionExample {

**public** **static** **void** main(String[] args) {

// Create the Hibernate SessionFactory

SessionFactory sessionFactory = **new** Configuration()

.configure("hibernate.cfg.xml")

.addAnnotatedClass(Author.**class**)

.addAnnotatedClass(Book.**class**)

.buildSessionFactory();

// Create some test data

*createTestData*(sessionFactory);

// Retrieve an Author and access their books lazily

Long authorId = 1L;

**try** (Session session = sessionFactory.openSession()) {

Author author = session.get(Author.**class**, authorId);

// At this point, only the Author entity is fetched, and books are not loaded yet.

**if** (author != **null**) {

List<Book> books = author.getBooks(); // Triggering lazy loading

// Now the books associated with the author are loaded from the database.

// Print the author's name and books

System.***out***.println("Author: " + author.getName());

System.***out***.println("Books: ");

**for** (Book book : books) {

System.***out***.println("- " + book.getTitle());

}

} **else** {

System.***out***.println("Author not found with ID: " + authorId);

}}

**catch** (Exception e) {

e.printStackTrace();

} **finally** {

// Clean up and close the SessionFactory

sessionFactory.close();

}

}

**private** **static** **void** createTestData(SessionFactory sessionFactory) {

**try** (Session session = sessionFactory.openSession()) {

session.beginTransaction();

// Create an Author

Author author = **new** Author("John Doe");

// Create some Books associated with the Author

Book book1 = **new** Book("Book 1", author);

Book book2 = **new** Book("Book 2", author);

// Save the Author and Books

session.save(author);

session.save(book1);

session.save(book2);

session.getTransaction().commit();

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"<http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd>">

<hibernate-configuration>

<session-factory>

<property name=*"hibernate.connection.driver\_class"*>com.mysql.jdbc.Driver</property>

<property name=*"hibernate.connection.password"*>Spidey@0987</property>

<property name=*"hibernate.connection.url"*>jdbc:mysql://localhost/mydatabase</property>

<property name=*"hibernate.connection.username"*>root</property>

<property name=*"hibernate.dialect"*>org.hibernate.dialect.MySQLDialect</property>

</session-factory>

</hibernate-configuration>

Output :

