

[Home](#) › [Interview](#) › [Spring, Hibernate, & Maven Interview Q&A](#) › [Hibernate](#) › 03:

Identifying and fixing NonUniqueObjectException in Hibernate

03: Identifying and fixing NonUniqueObjectException in Hibernate

Posted on [September 1, 2015](#) by [Arulkumaran Kumaraswamipillai](#)

NonUniqueObjectException is thrown when there is an object already associated with the session with the same id (primary key) (i.e. a duplicate). It is important to understand the concept of a “detached” object in Hibernate. This is one of the most common errors, understanding why and when this error is thrown will save you time in identifying and fixing this issue.

Q. What is a detached object in Hibernate?

600+ Full Stack Java/JEE Interview Q&As ♥Free ♦FAQs

[open all](#) | [close all](#)

[Ice Breaker Interview](#)

[Core Java Interview C](#)

[JEE Interview Q&A \(3](#)

[Pressed for time? Jav](#)

[SQL, XML, UML, JSC](#)

[Hadoop & BigData Int](#)

[Java Architecture Inte](#)

[Scala Interview Q&As](#)

[Spring, Hibernate, & I](#)

[Spring \(18\)](#)

[Hibernate \(13\)](#)

[01: ♥♦ 15+ Hiber](#)

[01b: ♦ 15+ Hiber](#)

[02: Understandir](#)

[03: Identifying ar](#)

[04: Identifying ar](#)

[05: Debugging H](#)

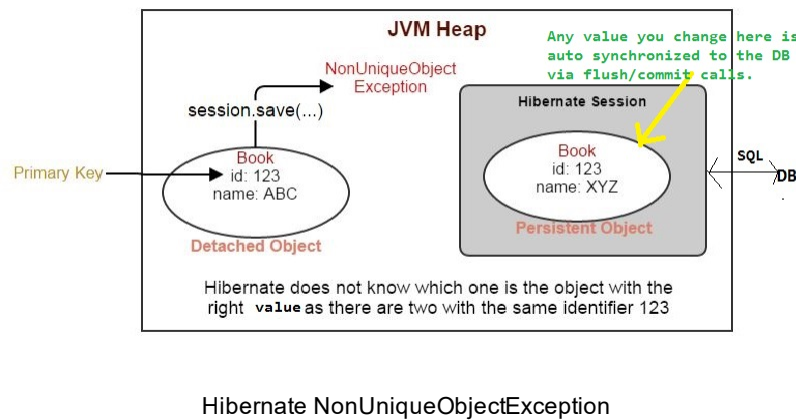
[06: Hibernate Fii](#)

[07: Hibernate mi](#)

[08: Hibernate au](#)

[09: Hibernate en](#)

[10: Spring, Java](#)



- 11: Hibernate de
- 12: Hibernate cu
- AngularJS (2)
- Git & SVN (6)
- JMeter (2)
- JSF (2)
- Maven (3)
- Testing & Profiling/Sa
- Other Interview Q&A 1
- Free Java Interview

When you close an individual Hibernate Session, the persistent objects you are working with become **detached**. This means the objects are still in the application's memory, but Hibernate is no longer responsible for tracking changes to those detached objects. If you then modify a detached object and want to update it, you have to reattach that object. During that reattachment process, Hibernate will verify if there are any other copies of the same object. If it finds any already in the session, it gets confused as to which one is the correct object, hence notifies you with a "NonUniqueObjectException" as opposed to saving any bad data.

"**LazyInitializationException**" is another common error due to a detached object. This error occurs when you try to access properties or associated object of Detached object then it results in Lazy Initialization Exception. This means Hibernate "could not initialize the proxy" when there is no Session".

Scenario 1: whilst update(..) or saveOrUpdate(..)

Step 1: Load an entity say a "Book" from the database into Hibernate session.

```
1 session.get(Book.class, new Integer(123));
```

16 Technical Key Areas

[open all](#) | [close all](#)

- Best Practice (6)
- Coding (26)
- Concurrency (6)
- Design Concepts (7)
- Design Patterns (11)
- Exception Handling (3)
- Java Debugging (21)
- Judging Experience I
- Low Latency (7)
- Memory Management
- Performance (13)
- QoS (8)
- Scalability (4)
- SDLC (6)
- Security (13)
- Transaction Managen

80+ step by step Java Tutorials

[open all](#) | [close all](#)

- Setting up Tutorial (6)
- Tutorial - Diagnosis (2)

Step 2: Convert the entity to a DTO, say “BookDto”, and pass it to populate the view

Step 3: Modify a value in the “BookDto”

```
1 myBookDto.setSomething(newValue);
```

Step 4: Convert back the “BookDto” to Hibernate entity “Book”.

Step 5: Try to update the new entity “Book”, and you will get “NonUniqueObjectException”

```
1 session.update(book); //NonUniqueObjectException
```

Fix:

1) Call **merge()** instead. Merge() takes the instance passed and merges it with any object of the “same id” associated with the session. Note, if you plan on using the object again you must use the object returned by the call to merge(). The instance object passed to merge will not be updated.

```
1 Object mergedObj = session.merge(book);
```

2) If it is possible, don’t convert the entity “Book” to DTO “BookDto”, and “setSomething” on the entity itself. You don’t have to update the entity “Book” as it is already being maintained by Hibernate. Later flush() will push any changes recorded in the session to the database.

Scenario 2: whilst delete(...)

Even if you try to delete the entity “Book” from the session, you will get the “NonUniqueObjectException”. This is because Hibernate uses the object “**identity equals**” where you have two different physical “Book” objects in the JVM Heap, but logically both are the same objects with the same identifier,

- [Akka Tutorial \(9\)](#)
- [Core Java Tutorials \(2\)](#)
- [Hadoop & Spark Tuto](#)
- [JEE Tutorials \(19\)](#)
- [Scala Tutorials \(1\)](#)
- [Spring & Hibernate T](#)
- [Tools Tutorials \(19\)](#)
- [Other Tutorials \(45\)](#)

100+ Java pre-interview coding tests

[open all](#) | [close all](#)

- [Can you write code? \(1\)](#)
- [♦ Complete the given](#)
- [Converting from A to I](#)
- [Designing your classe](#)
- [Java Data Structures](#)
- [Passing the unit tests](#)
- [What is wrong with th](#)
- [Writing Code Home A](#)
- [Written Test Core Jav](#)
- [Written Test JEE \(1\)](#)

How good are your?

[open all](#) | [close all](#)

- [Career Making Know-](#)
- [Job Hunting & Resum](#)

say “id = 123”, which corresponds to the primary key in the database table.

Step 1: Load a “Book” from the database, and assign it to the reference “bookXyz”;

```
1 Book bookXyz = session.get(Book.class, new Integer(123));
```

Step 2: Instantiated the same “Book” object again later, and assign it to the reference “bookXyzAgain”.

```
1 Book bookXyzAgain = new Book(123);
```

Step 3: You try to delete “bookXyzAgain” from the session, and you will get “NonUniqueObjectException”.

```
1 session.delete(bookXyzAgain); //NonUniqueObjectException
```

The exception is thrown because you are trying to delete an object which has the same primary key as “bookXyz”, and it already exists in the hibernate session.

Fix:

```
1 Object mergedBook=session.merge(bookXyzAgain);  
2 session.delete(mergedBook);
```

Scenario 3: whilst cascade saves

When there is a cascade save between two object instances say “a” of type “Book” and “b” of type “Author”, but the object instance of “Author” has already been associated with the session but not on the same instance of b, but instance of “c”, then you can get this exception.

A **circular save** request can throw this exception as well. For example, if you have set “**cascade=all**” on both ends of a one-to-many relationship resulting in a circular save request.

General tips to fix these types of Hibernate exceptions

These errors can be sometimes hard to debug,

1. You need to break down your code, and then comment out some parts until the error goes away and then put the code back until the error comes back again. Then you know what is causing it.
2. Check if you had forgotten to put @GeneratedValue for @Id column in your hibernate entity.

Popular Member Posts

♦ 11 Spring boot interview questions & answers

904 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

816 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

427 views

18 Java scenarios based interview Questions and Answers

408 views

♦ 7 Java debugging interview questions & answers

323 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

311 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

303 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

301 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

251 views

001B: ♦ Java architecture & design concepts

interview questions & answers

209 views

10

Like

Share

Tweet

↑

submit

↓

reddit

4

G+1

12

Share

Bio

Latest Posts

**Arulkumaran Kumaraswamipillai**

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. [join my LinkedIn Group](#). [Reviews](#)

**About Arulkumaran Kumaraswamipillai**

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. [join my LinkedIn Group](#). [Reviews](#)

[← Basic Apache Camel and Spring tutorial](#)1. Asynchronous processing in Java real life examples – part-1 [→](#)

Posted in Hibernate, Java Debugging, member-paid

Tags: Free Content

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ [Java generics in no time](#) ☀ [Top 6 tips to transforming your thinking from OOP to FP](#) ☀ [How does a HashMap internally work? What is a hashing function?](#)

☀ [10+ Java String class interview Q&As](#) ☀ [Java auto un/boxing benefits & caveats](#) ☀ [Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect](#)

Non-Technical Know Hows

☀ [6 Aspects that can motivate you to fast-track your career & go places](#) ☀ [Are you reinventing yourself as a Java developer?](#) ☀ [8 tips to safeguard your Java career against offshoring](#) ☀ [My top 5 career mistakes](#)

Prepare to succeed

☀ [Turn readers of your Java CV go from “Blah blah” to “Wow”?](#) ☀ [How to prepare for Java job interviews?](#) ☀ [16 Technical Key Areas](#) ☀ [How to choose from multiple Java job offers?](#)

Select Category ▼

© Disclaimer

The contents in this Java-Success are copy righted. The author has the right to correct or enhance the current content without any prior notice.

These are general advice only, and one needs to take his/her own circumstances into consideration. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. No guarantees are made regarding the accuracy or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.