

[Home](#) › [Interview](#) › [Spring, Hibernate, & Maven Interview Q&A](#) › [Hibernate](#) › 04:

Identifying and fixing LazyInitializationException in Hibernate

# 04: Identifying and fixing LazyInitializationException in Hibernate

Posted on [September 3, 2015](#) by [Arulkumaran Kumaraswamipillai](#)

**LazyInitializationException** is thrown when an object becomes detached, and if you try to access associated (i.e. proxied) object(s) of a detached object.

## Q. What is a detached object in Hibernate?

When you close an individual Hibernate Session, the persistent objects you are working with are **detached**. This means the object is still in the application's memory, but Hibernate is no longer responsible for tracking changes to that object. If your detached object has any associated objects, for example a "Book" object might have an associated proxied "Author" object. So, when you do something like

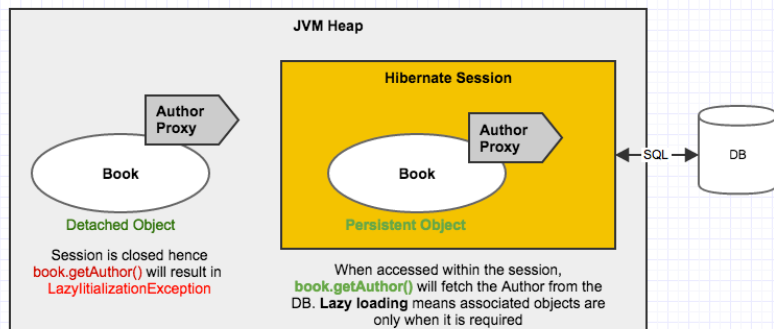
## 600+ Full Stack Java/JEE Interview Q&As ♥Free ♦FAQs

[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

```
1 myBook.getAuthor().getName(); //needs to go to th
```

Hibernate needs to go database to initialize the proxy object. This is known as the “lazy” initialization of the “Author” object. Load it only when it is required. Now, if your Hibernate session is closed, then the database connection is closed and Hibernate cannot load the proxy object.



Hibernate Lazy InitializationException

## Q. What is a unit-of-work?

A session means a physical connection to a database. Hibernate session is a **unit of work**. You start a unit of work by “opening a session” and close the unit of work by closing a session. You also **flush()** a Session at the end of a unit of work to execute the SQL. The right approach to manage a “unit of work” is to define clear transaction boundaries in your application by beginning and committing transactions either declaratively or programmatically.

## Session-per-request Vs long-conversations unit-of-work

The most common pattern in a multi-user client/server application is **session-per-request**. In this model, a request from the client is sent to the server, where the Hibernate persistence layer runs. A new Hibernate Session is opened, and all database operations are executed in this unit of work. On completion of the work, and once the response

- 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

## 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 Managemen
- 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)

for the client has been prepared, the session is flushed or committed and the session is closed.

```

1 Session session = factory.openSession();
2 Transaction tx = null;
3 try {
4     tx = session.beginTransaction();
5     // do some Create, Read, Update, Delete work.
6     ...
7     tx.commit(); //write to db
8 }
9 catch (Exception e) {
10     if (tx!=null) tx.rollback();
11     e.printStackTrace();
12 } finally {
13     session.close();
14 }

```

A very common approach is to use Spring with Hibernate, and Spring will take care of the transaction management. Start a transaction when a server request has to be processed, and end the transaction before the response is sent to the client. Add Spring @Transactional annotation on the service layer making calls to the Hibernate repository or DAO layer. Spring provides a transaction abstraction layer over the Hibernate transaction API, and enables persistent operations to participate in global transactions.

“org.springframework.orm.hibernate3.HibernateTransactionManager” for the local transactions (i.e. 1 datasource) and org.springframework.transaction.jta.JtaTransactionManager for the global JTA transactions.

There are business usecases that require a “conversational scope” with the user that are interleaved with multiple database accesses. This requires **Long conversations**











- 1) User retrieves data
- 2) The user modifies some objects and clicks save after 2 minutes.
- 3) The user makes more changes ....

You have to use several database transactions to implement this conversation. It is a bad idea to keep the transaction open for the whole conversation, even during the user think time (e.g. 2 minute before saving the loaded data). So, to

-  [Akka Tutorial \(9\)](#)
-  [Core Java Tutorials \(2\)](#)
-  [Hadoop & Spark Tuto](#)
-  [JEE Tutorials \(19\)](#)
-  [Scala Tutorials \(1\)](#)
-  [Spring & Hlberate Ti](#)
-  [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](#)

deal with these types of long conversations Hibernate has features such as

**1) Detached Objects:** Allowing you to apply the **session-per-request** pattern where all the loaded instances will be “**detached**” during the user think time, and Hibernate allows you to reattach them to a new session. This pattern is known as **session-per-request-with-detached-objects**.

**2) Extended (or Long) Session:** where the Hibernate Session can be disconnected from the underlying JDBC connection after the database transaction has been committed and reconnected when a new client request occurs. This pattern is known as **session-per-conversation** and makes even reattachment unnecessary.

In both the above patterns, the Hibernate’s **automatic Versioning** can be used to detect and prevent possibilities of any **concurrent modification** issues due to the use think times. For example, users A and B might view the same data and user “A” might modify & save after “1 minute” of think time and user “B” might try to save after “2 minute” of think time not being aware of user “A’s” changes. The Hibernate versioning will indicate that the user “B’s” data is stale by throwing an exception where the user “B” has to redo the modification on the modified data.

## Q. When is this exception thrown?

**Step 1:** User retrieves data

```
1 //load the persistent object from database
2 Book book = session.get(Book.class, new Integer(12))
```

**Step 2:** The Hibernate session gets closed, as the user may take some time to modify the data or drill down into author details. You don’t want to keep the session open during the think time.

**Step 3:** The user drills down into the loaded “Book” object to view the “Author” details. The “Book” is a detached object.

```
1 //book is a detached object.  
2 String authName = book.getAuthor().getName(); //L
```

Now the “LazyInitializationException” is thrown as the Hibernate session is closed and cannot load the proxied “Author” data.

## Q. How to fix this exception?

**Approach 1:** Eagerly fetch the Author without proxying

```
1 @OneToOne(fetch = FetchType.EAGER)  
2 @JoinTable(name = 'author')  
3 private Author author;
```

This approach is easy to setup, but if you have a complex object dependency graph, you will end up eagerly loading lots of objects into the heap, and causing performance issues.

**Approach 2:** Hibernate extended session

If you are using JPA with Hibernate, you can use the `@PersistenceContext(type = PersistenceContextType.EXTENDED)`. This allows you to access a lazy property anywhere (i.e. even if it is detached). `PersistenceContextType.EXTENDED` means that you are in charge of managing your session, and Spring will not do it for you. `PersistenceContextType.EXTENDED` is useful in limited scenarios where a **session-per-conversation** is required.

You need to be very careful with this approach as it can cause “N+1” issues.

**Approach 3:** Use the join query

```
1 Query query = entityManager.createQuery('select b
```

This is more like fetching eagerly on a as needed basis.

## Approach 4: Load or get the entity again

Load the associated entity within the same session.

```
1  ...
2  try {
3      Book book = session.get(Book.class, new Integer(id));
4      Author author = book.getAuthor();
5      return author;
6  } catch (Exception e) {
7      // ...
8  } finally {
9      //close the session
10 }
```

## 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

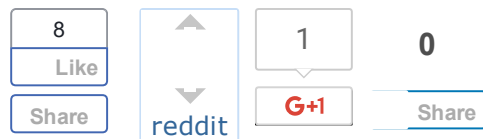
♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

251 views

## 001B: ♦ Java architecture &amp; design concepts

## interview questions &amp; answers

209 views



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](https://www.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](https://www.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](#)

[◀ 1. Asynchronous processing in Java real life examples – part-1](#)[05: Debugging Hibernate & getting a better handle on the 4 most common issues ▶](#)

**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.