* [Home](https://www.javainuse.com/home)
* [Java](https://www.javainuse.com/)
* [Spring](https://www.javainuse.com/)
* [Angular](https://www.javainuse.com/)
* [Full Stack](https://www.javainuse.com/)
* [Apache Camel](https://www.javainuse.com/)
* [Cloud Frameworks](https://www.javainuse.com/)
* [Messaging](https://www.javainuse.com/)
* [Drools](https://www.javainuse.com/)
* [Search Engine](https://www.javainuse.com/)
* [Hazelcast](https://www.javainuse.com/hazelcast)
* [DevOps](https://www.javainuse.com/)

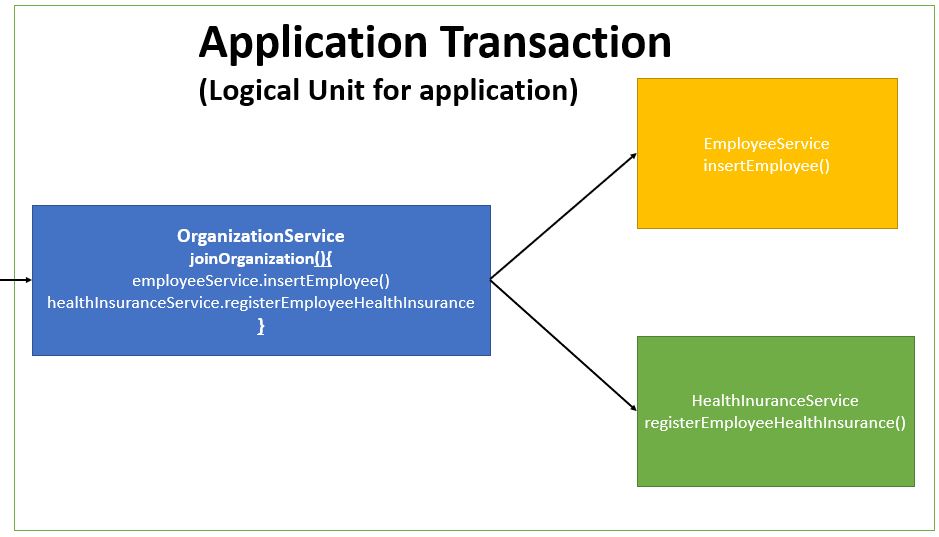
Search Tutorials

Top of Form

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | |  |  | |  |

Bottom of Form

Spring Boot Transaction Management - Transaction Rollback Example

In a previous tutorials we had implemented [Spring Boot Transaction Management Example](https://www.javainuse.com/spring/boot-transaction) - we saw what are transactions and implemented declarative transaction management. In a previous tutorial - [Spring Boot Transactions - Understanding Transaction Propagation](https://www.javainuse.com/spring/boot-transaction-propagation) we also looked at what is propagation and its different types using Spring Boot. In next tutorial we will be looking at [what are Transaction Isolations using Spring Boot.](https://www.javainuse.com/spring/boot-transaction-isolation) In [Spring Boot Transaction Management Example](https://www.javainuse.com/spring/boot-transaction) we had seen application transaction is a sequence of application actions that are considered as a single logical unit by the application.  
  
For an application transaction if any action fails then all other actions gets rolled back. Previous [Transaction Management Example](https://www.javainuse.com/spring/boot-transaction) we had tested the rollback by throwing an unchecked exception.  
**However in real time scenarios it is the checked exception that gets thrown. These are business exceptions based on some logic**  
So how will our transactions behave in case of Checked Exceptions? In case of **checked exceptions the previously executed transactions do not get rolled back automatically even if we have used transaction annotation.** We need to inform the application how to handle roll back in event of checked exception. This is achieved using the **RollbackFor** annotation.

Spring Boot Transaction Management - Table of Contents

[Spring Boot Transaction Management Example](https://www.javainuse.com/spring/boot-transaction)

[Spring Boot Transactions - Understanding Transaction Propagation](https://www.javainuse.com/spring/boot-transaction-propagation)

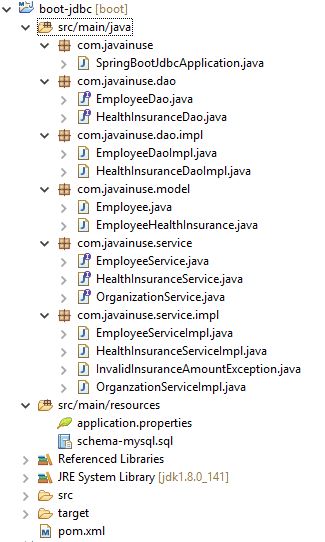
[**Spring Boot Transactions - Understanding Transaction Rollbacks**](https://www.javainuse.com/spring/boot-rollback)

[Spring Boot Transactions - Understanding Transaction Isolation](https://www.javainuse.com/spring/boot-transaction-isolation)

Video

This tutorial is explained in the below Youtube Video.

Lets Begin-

We will be modifying the code we had [created previously for Transaction Management.](https://www.javainuse.com/spring/boot-transaction)  
We will create a custom checked exception called **InvalidInsuranceAmountException.** According to our business logic if the insurance coverage amount is less than zero then this exception should get thrown. The maven project will be as follows -  
  
Create the Custom Exception as follows-

package com.javainuse.service.impl;

public class InvalidInsuranceAmountException extends Exception {

private static final long serialVersionUID = 1L;

public InvalidInsuranceAmountException(String cause) {

super(cause);

}

}

The HealthInsuranceService interface throws this exception for the registerEmployeeHealthInsurance method.

package com.javainuse.service;

import com.javainuse.model.EmployeeHealthInsurance;

import com.javainuse.service.impl.InvalidInsuranceAmountException;

public interface HealthInsuranceService {

void registerEmployeeHealthInsurance(EmployeeHealthInsurance employeeHealthInsurance)

**throws InvalidInsuranceAmountException**;

void deleteEmployeeHealthInsuranceById(String empid);

}

In the HealthInsuranceServiceImpl class for the registerEmployeeHealthInsurance we put a check for verifying if the coverage amount is less than zero. If it is then we throw the InvalidInsuranceAmountException.

package com.javainuse.service.impl;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.javainuse.dao.HealthInsuranceDao;

import com.javainuse.model.EmployeeHealthInsurance;

import com.javainuse.service.HealthInsuranceService;

@Service

@Transactional

public class HealthInsuranceServiceImpl implements HealthInsuranceService {

@Autowired

HealthInsuranceDao healthInsuranceDao;

@Override

public void registerEmployeeHealthInsurance(EmployeeHealthInsurance employeeHealthInsurance)

throws InvalidInsuranceAmountException {

**if (employeeHealthInsurance.getCoverageAmount() < 0) {**

**throw new InvalidInsuranceAmountException("Coverage Amount Should not be negative");**

**}**

healthInsuranceDao.registerEmployeeHealthInsurance(employeeHealthInsurance);

}

@Override

public void deleteEmployeeHealthInsuranceById(String empid) {

healthInsuranceDao.deleteEmployeeHealthInsuranceById(empid);

}

}

The OrganizationService interface for the joinOrganization method we throw the InvalidInsuranceAmountException.

package com.javainuse.service;

import com.javainuse.model.Employee;

import com.javainuse.model.EmployeeHealthInsurance;

import com.javainuse.service.impl.InvalidInsuranceAmountException;

public interface OrganizationService {

public void joinOrganization(Employee employee, EmployeeHealthInsurance employeeHealthInsurance)

**throws InvalidInsuranceAmountException;**

public void leaveOrganization(Employee employee, EmployeeHealthInsurance employeeHealthInsurance);

}

In the OrganzationServiceImpl which makes a call to the HealthInsuranceService class, we catch the InvalidInsuranceAmountException log it and throw it again. This is because we want the calling client to know what exception has occurred.

package com.javainuse.service.impl;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.javainuse.model.Employee;

import com.javainuse.model.EmployeeHealthInsurance;

import com.javainuse.service.EmployeeService;

import com.javainuse.service.HealthInsuranceService;

import com.javainuse.service.OrganizationService;

@Service

@Transactional

public class OrganzationServiceImpl implements OrganizationService {

@Autowired

EmployeeService employeeService;

@Autowired

HealthInsuranceService healthInsuranceService;

@Override

@Transactional

public void joinOrganization(Employee employee, EmployeeHealthInsurance employeeHealthInsurance)

**throws InvalidInsuranceAmountException** {

employeeService.insertEmployee(employee);

try {

healthInsuranceService.registerEmployeeHealthInsurance(employeeHealthInsurance);

} **catch (InvalidInsuranceAmountException e) {**

**throw new InvalidInsuranceAmountException("Exception is thrown");**

**}**

}

@Override

public void leaveOrganization(Employee employee, EmployeeHealthInsurance employeeHealthInsurance) {

employeeService.deleteEmployeeById(employee.getEmpId());

healthInsuranceService.deleteEmployeeHealthInsuranceById(employeeHealthInsurance.getEmpId());

}

}

Finally in the SpringBootJdbcApplication we throw the InvalidInsuranceAmountException.

package com.javainuse;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import com.javainuse.model.Employee;

import com.javainuse.model.EmployeeHealthInsurance;

import com.javainuse.service.EmployeeService;

import com.javainuse.service.OrganizationService;

import com.javainuse.service.impl.InvalidInsuranceAmountException;

@SpringBootApplication

public class SpringBootJdbcApplication {

@Autowired

EmployeeService employeeService;

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

ApplicationContext context = SpringApplication.run(SpringBootJdbcApplication.class, args);

OrganizationService organizationService = context.getBean(OrganizationService.class);

Employee emp= new Employee();

emp.setEmpId("emp1");

emp.setEmpName("emp1");

EmployeeHealthInsurance employeeHealthInsurance= new EmployeeHealthInsurance();

employeeHealthInsurance.setEmpId("emp1");

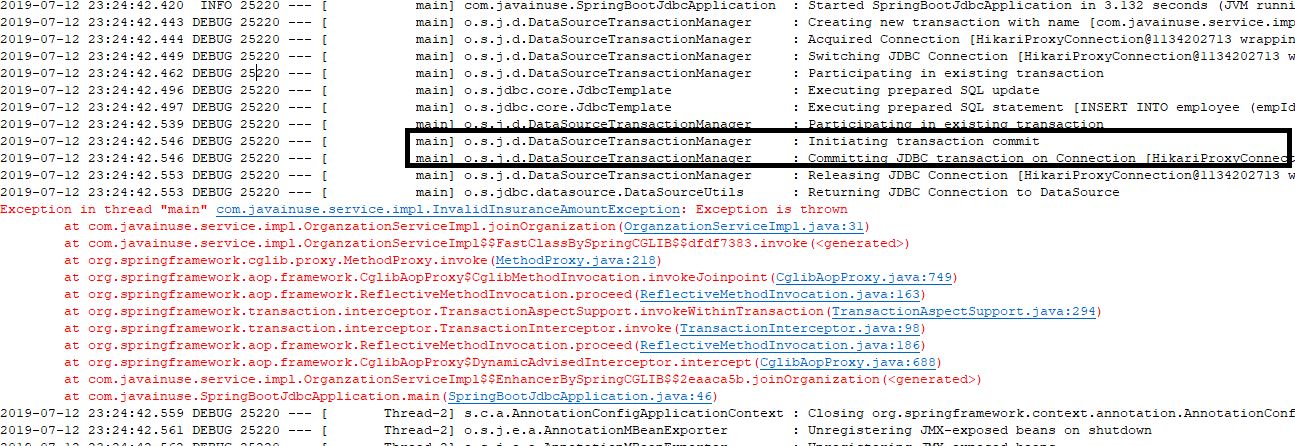
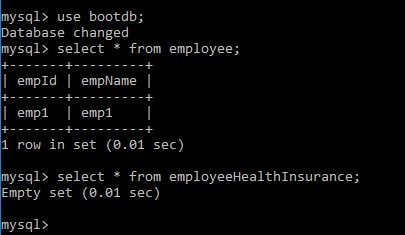
employeeHealthInsurance.setHealthInsuranceSchemeName("premium");

employeeHealthInsurance.setCoverageAmount(0);

organizationService.joinOrganization(emp, employeeHealthInsurance);

}

}

If now run the application- We see that the employeeService transaction is not rolled back due to an exception in employeeHealthService.  
  
In the Database we see that the insert for employee table has not been rolledback-  
  
But this should not be the case. To achieve roll back for checked exception we will need to specify it using Rollbackfor Annotation.

package com.javainuse.service.impl;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.javainuse.model.Employee;

import com.javainuse.model.EmployeeHealthInsurance;

import com.javainuse.service.EmployeeService;

import com.javainuse.service.HealthInsuranceService;

import com.javainuse.service.OrganizationService;

@Service

@Transactional

public class OrganzationServiceImpl implements OrganizationService {

@Autowired

EmployeeService employeeService;

@Autowired

HealthInsuranceService healthInsuranceService;

@Override

@Transactional(**rollbackFor = InvalidInsuranceAmountException.class**)

public void joinOrganization(Employee employee, EmployeeHealthInsurance employeeHealthInsurance)

throws InvalidInsuranceAmountException {

employeeService.insertEmployee(employee);

try {

healthInsuranceService.registerEmployeeHealthInsurance(employeeHealthInsurance);

} catch (InvalidInsuranceAmountException e) {

throw new InvalidInsuranceAmountException("Exception is thrown");

}

}

@Override

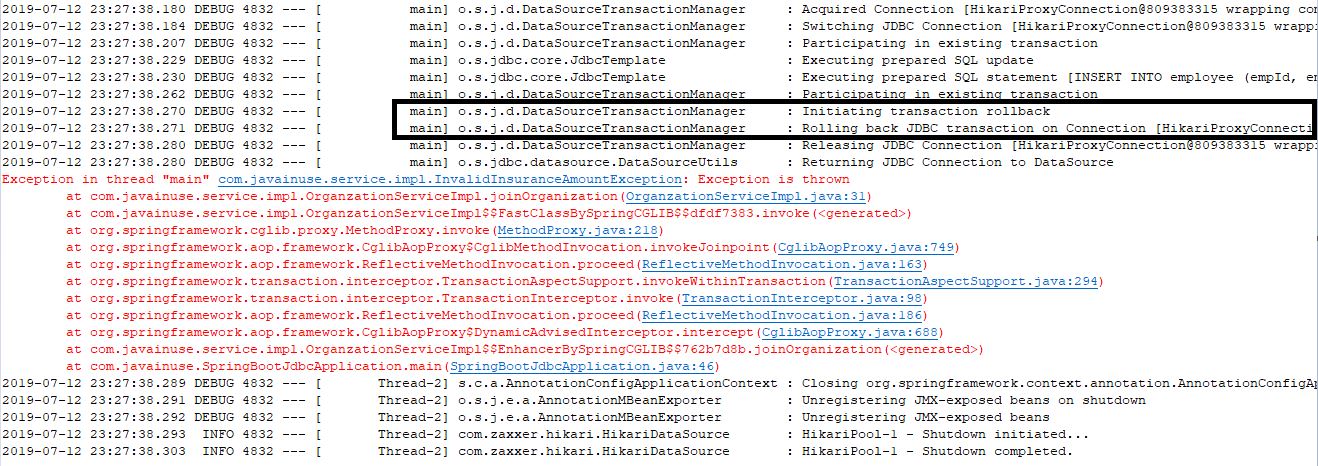
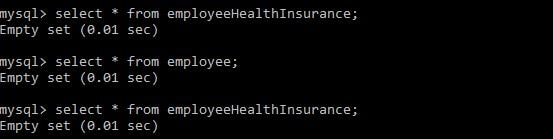
public void leaveOrganization(Employee employee, EmployeeHealthInsurance employeeHealthInsurance) {

employeeService.deleteEmployeeById(employee.getEmpId());

healthInsuranceService.deleteEmployeeHealthInsuranceById(employeeHealthInsurance.getEmpId());

}

}

Now run the application again. We see that the employeeService transaction is rolled back due to an exception in employeeHealthService.  
  
In the Database we see that the insert for employee table has been rolledback-  


Download Source Code

Download it -  
[Spring Boot Transaction Management Rollback](https://www.javainuse.com/zip/spring/boot/boot-jdbc-rollback.rar)

Popular Posts

* [E-commerce Website - Online Book Store using Angular 8 + Spring Boot](https://www.javainuse.com/fullstack/ecommerce)
* [Spring Boot +JSON Web Token(JWT) Hello World Example](https://www.javainuse.com/spring/boot-jwt)
* [Angular 7 + Spring Boot Application Hello World Example](https://www.javainuse.com/spring/ang7-hello)
* [Build a Real Time Chat Application using Spring Boot + WebSocket + RabbitMQ](https://www.javainuse.com/spring/boot-websocket-chat)
* [Pivotal Cloud Foundry Tutorial - Deploy Spring Boot Application Hello World Example](https://www.javainuse.com/pcf/pcf-hello)
* [Deploying Spring Based WAR Application to Docker](https://www.javainuse.com/devOps/docker/docker-war)
* [EIP patterns using Apache Camel](https://www.javainuse.com/camel/camel_EIP)
* [Spring Cloud- Netflix Eureka + Ribbon Simple Example](https://www.javainuse.com/spring/spring_ribbon)
* [Spring Cloud- Netflix Hystrix Circuit Breaker Simple Example](https://www.javainuse.com/spring/spring_hystrix_circuitbreaker)
* [Spring Boot + Swagger Example Hello World Example](https://www.javainuse.com/spring/boot_swagger)
* [Spring Boot Batch Simple example](https://www.javainuse.com/spring/bootbatch)
* [Spring Boot + Apache Kafka Example](https://www.javainuse.com/spring/spring-boot-apache-kafka-hello-world)
* [Spring Boot Admin Simple Example](https://www.javainuse.com/spring/boot-admin)
* [Spring Boot Security - Introduction to OAuth](https://www.javainuse.com/spring/spring-boot-oauth-introduction)
* [Spring Boot OAuth2 Part 1 - Getting The Authorization Code](https://www.javainuse.com/spring/spring-boot-oauth-authorization-code)
* [Spring Boot OAuth2 Part 2 - Getting The Access Token And Using it to Fetch Data.](https://www.javainuse.com/spring/spring-boot-oauth-access-token)
* [JBoss Drools Hello World-Stateful Knowledge Session using KieSession](https://www.javainuse.com/drools_hello_kie)
* [Understand Drools Stateful vs Stateless Knowledge Session](https://www.javainuse.com/drools_states)
* [JBoss Drools- Understanding Drools Decision Table using Simple Example](https://www.javainuse.com/drools/drools_decision)

See Also

* [Spring Boot Interview Questions](https://www.javainuse.com/spring/SpringBootInterviewQuestions)
* [Spring Batch Interview Questions](https://www.javainuse.com/spring/sprbatch_interview)
* [Spring AOP Interview Questions](https://www.javainuse.com/spring/spring-AOP-interview-quesions)
* [Angular 2 Interview Questions](https://www.javainuse.com/angular/ang2_intvw)
* [Apache Camel Interview Questions](https://www.javainuse.com/camel/Apache_Camel_Questions)
* [JBoss Fuse Interview Questions](https://www.javainuse.com/camel/JBoss_Fuse_Questions)
* [Drools Interview Questions](https://www.javainuse.com/drools/drools_intvw)
* [Java 8 Interview Questions](https://www.javainuse.com/java/java8_intvw)
* [Spring Cloud Interview Questions](https://www.javainuse.com/spring/spring-cloud-interview-questions)
* [Microservices Interview Questions](https://www.javainuse.com/spring/microservices-interview-quesions)
* [Java HashMap and ConcurrentHashMap Interview Questions](https://www.javainuse.com/java/java_map_intvw)
* [Mule ESB frequently asked interview questions](https://www.javainuse.com/misc/muleintvw)
* [Apache Kafka Interview Questions](https://www.javainuse.com/misc/apache-kafka-interview-questions)
* [Tosca Testing Tool Interview Questions](https://www.javainuse.com/misc/tosca-testing-tool-interview-questions)
* [Top Maven Build Tool Interview Questions](https://www.javainuse.com/misc/maven-interview-questions)
* [Top Gradle Build Tool Interview Questions](https://www.javainuse.com/misc/gradle-interview-questions)
* [Miscellaneous Topics](https://www.javainuse.com/misc)

Ad by Valueimpression

© Copyright JavaInUse. All Rights Reserved.  
[Privacy Policy](https://www.javainuse.com/privacy)