load() and get() difference in Hibernate:

Example:

1. Pom.xml file:

```
<properties>
		<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
		<maven.compiler.source>15</maven.compiler.source>
		<maven.compiler.target>15</maven.compiler.target>
	</properties>

	<dependencies>
		<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
		<dependency>
			<groupId>org.projectlombok</groupId>
			<artifactId>lombok</artifactId>
			<version>1.18.18</version>
			<scope>provided</scope>
		</dependency>

		<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
		<dependency>
			<groupId>mysql</groupId>
			<artifactId>mysql-connector-java</artifactId>
			<version>8.0.22</version>
		</dependency>

		<dependency>
			<groupId>junit</groupId>
			<artifactId>junit</artifactId>
			<version>4.11</version>
			<scope>test</scope>
		</dependency>
		<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
		<dependency>
			<groupId>org.hibernate</groupId>
```

```
            <artifactId>hibernate-core</artifactId>
            <version>5.4.27.Final</version>
        </dependency>
        <!--
https://mvnrepository.com/artifact/com.jslsolucoes/ojdbc6 -->
        <dependency>
            <groupId>com.jslsolucoes</groupId>
            <artifactId>ojdbc6</artifactId>
            <version>11.2.0.1.0</version>
        </dependency>
        </dependencies>
```

2. Configuration file:

```
<?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>

        <!-- Connection Properties -->
        <property
name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property
name="connection.url">jdbc:mysql://localhost:3306/webservices</proper
ty>
        <property name="connection.username">root</property>
        <property name="connection.password">root</property>

        <!-- dialect,show sql,format properties -->
        <property
name="dialect">org.hibernate.dialect.MySQL55Dialect</property>
        <property name="show_sql">true</property>
        <property name="format_sql">true</property>

        <!-- Table Creation and Updation properties -->
        <property name="hbm2ddl.auto">update</property>

        <!-- Model class linking -->
        <mapping class="in.nit.model.Company"/>

        </session-factory>
```

```
</hibernate-configuration>
```

3. Model class Company

```
package in.nit.model;

import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

import lombok.Data;
@SuppressWarnings("serial")
@Data
@Entity
//if not mention @table then Model class as table first letter small
public class Company  implements Serializable{
	@Id
	@GeneratedValue(strategy = GenerationType.SEQUENCE)
	private Integer cmpId;
	private String cmpName;
	private String cmpContact;

}
```

4. Hibernate Util

```
package in.nit.util;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;


public class HibernateUtil {

private static SessionFactory sf;

static {
	try {
```

```
			sf=new
Configuration().configure("/in/nit/cfgs/hibernate.cfg.xml").buildSessionFa
ctory();
		} catch (Exception e) {
			e.printStackTrace();
		}
}

		public static SessionFactory getSf() {
			return sf;
}

}
```

5. SaveTest:

```
package in.nit.test;
import org.hibernate.Session;
import org.hibernate.Transaction;

import in.nit.model.Company;
import in.nit.util.HibernateUtil;

public class SaveTest {
public static void main(String[] args) {
		Session ses=HibernateUtil.getSf().openSession();
		Transaction tx=null;
		try(ses){
			tx=ses.beginTransaction();
			//save Record into db
			Company c=new Company();
			c.setCmpName("Wipro");
			c.setCmpContact("wipro@gmail.com");
			ses.save(c);
			System.out.println("Record Saved ");
			tx.commit();
		}
		catch(Exception e) {
			if(tx!=null && tx.getStatus().canRollback()) {
				tx.rollback();
			e.printStackTrace();
			System.out.println("Record Not  Saved ");
			}
		}
```

```
}
}
```

6. LoadTest:

```
package in.nit.test;
import java.util.Scanner;

import org.hibernate.Session;
import org.hibernate.Transaction;

import in.nit.model.Company;
import in.nit.util.HibernateUtil;

public class LoadTest {
public static void main(String[] args) {
	Session ses=HibernateUtil.getSf().openSession();
	Transaction tx=null;
	try(ses){
		tx=ses.beginTransaction();
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter ID Integer Value: ");
		int id=sc.nextInt();
		Company c=ses.load(Company.class,id);
		System.out.println("Fetch data"+c);
		tx.commit();
		sc.close();
	}
	catch(Exception e) {
		if(tx!=null && tx.getStatus().canRollback()) {
			tx.rollback();
		e.printStackTrace();
		System.out.println("Failed to fetch data");
		}
	}
}
}
```

7. GetTest:

```
package in.nit.test;
import java.util.Scanner;

import org.hibernate.Session;
```

```
import org.hibernate.Transaction;

import in.nit.model.Company;
import in.nit.util.HibernateUtil;

public class GetTest {
public static void main(String[] args) {
	Session ses=HibernateUtil.getSf().openSession();
	Transaction tx=null;
	try(ses){
		tx=ses.beginTransaction();
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter ID Integer Value: ");
		int id=sc.nextInt();
		Company c=ses.get(Company.class,id);
		System.out.println("Fetch data"+c);
		tx.commit();
		sc.close();
	}
	catch(Exception e) {
		if(tx!=null && tx.getStatus().canRollback()) {
			tx.rollback();
		e.printStackTrace();
		System.out.println("Failed to fetch data");
		}
	}
}
}
```

---

	load()	get()
	---	---
Opration:	Used for Fetch Data	Used for Fetch Data
Record not Found:	If Record not Fount getting NULL	If Record not Found getting Exception
Performance:	slower	slightly faster
Lazy Loading	it return Proxy Object[Fake Object] will not hit Database Immediately	No
Eager Loading	No	Will hit Database Immediately
Use Case:	If you not known ID value exist or not	If you known ID value exist

---