STOCK MANAGEMENT USING HIBERNATE

SOLUTION :

```
package org.model;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.List;

import javax.persistence.ElementCollection;
import javax.persistence.Query;

import org.dom4j.Branch;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class Solution1 {
	@ElementCollection

	private static void save(Stock stock) {
		SessionFactory sf = new
Configuration().configure().buildSessionFactory();
		Session session = sf.openSession();
		session.beginTransaction();
		session.save(stock);
		session.getTransaction().commit();
		session.close();
	}

	private static void addProduct() throws IOException {
		BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
		Stock stock = new Stock();
		System.out.println("PRODUCT NAME :");
		stock.setProductName(br.readLine());
		System.out.println("QUANTITY :");
		stock.setQuantity(Integer.valueOf(br.readLine()));
		System.out.println("PRICE :");
		stock.setPrice(Float.valueOf(br.readLine()));
		System.out.println("MANUFACTURE DATE :");
		stock.setManufactureDate(br.readLine());
		save(stock);
	}

	private static void viewAll() {
		SessionFactory sf = new
Configuration().configure().buildSessionFactory();
		Session session = sf.openSession();
		Query query = session.createQuery("from Stock");
		List<Stock> list = query.getResultList();
		for(Stock each:list) {
			System.out.println(each);
		}
	}

	private static void updateProduct() throws NumberFormatException,
IOException {
```

```
		BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
		SessionFactory sf = new
Configuration().configure().buildSessionFactory();
		Session session = sf.openSession();
		System.out.println("Enter Product ID:");
		int pid = Integer.valueOf(br.readLine());
		try {
			Stock stock = session.get(Stock.class, pid);
			System.out.println("PRODUCT NAME :");
			stock.setProductName(br.readLine());
			System.out.println("QUANTITY :");
			stock.setQuantity(Integer.valueOf(br.readLine()));
			System.out.println("PRICE :");
			stock.setPrice(Float.valueOf(br.readLine()));
			System.out.println("MANUFACTURE DATE :");
			stock.setManufactureDate(br.readLine());
			save(stock);
		}
		catch(Exception e) {
			System.out.println("PRODUCT ID NOT AVAILABLE");
		}

	}

	private static void deleteProduct() throws NumberFormatException,
IOException {
		BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
		SessionFactory sf = new
Configuration().configure().buildSessionFactory();
		Session session = sf.openSession();
		System.out.println("Enter Product ID (to delete):");
		int pid = Integer.valueOf(br.readLine());
		try {
			Stock stock = session.get(Stock.class, pid);
			session.delete(stock);
			save(stock);
		}
		catch(Exception e) {

		}
	}

	public static void main(String[] args) throws NumberFormatException,
IOException {
		BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
		System.out.println("********* STOCK MANAGEMENT *********");
		Login login = new Login();
		Authenticate au = new Authenticate();
		System.out.println("USERNAME :");
		login.setUserName(br.readLine());
		System.out.println("PASSWORD :");
		login.setPassword(br.readLine());
		if (!au.checkUser(login)) {
			System.out.println("INVALID USER");
			System.exit(0);
		}
```

```
		while(true) {
			System.out.println("1.Add Product\n2.View All\n3.Update
Product\n4.Delete Product\n5.Exit");
			switch(Integer.valueOf(br.readLine())) {
			case 1: addProduct();
				break;
			case 2: viewAll();
				break;
			case 3: updateProduct();
				break;
			case 4: deleteProduct();
				break;
			case 5: System.out.println("***BYE***");
				System.exit(0);
				break;
			default:System.out.println("Invaid Input");

			}
		}
	}

}
```

LOGIN :

```
package org.model;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Login {

	@Id
	private String userName;
	private String password;

	public String getUserName() {
		return userName;
	}
	public void setUserName(String userName) {
		this.userName = userName;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}

}
```

AUTHENTICATE :

```
package org.model;
```

```java
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class Authenticate {

	public boolean checkUser(Login user) {
		SessionFactory sf = new
Configuration().configure().buildSessionFactory();
		Session session = sf.openSession();
		session.beginTransaction();
		try {
		Login l = session.get(Login.class,user.getUserName());

	//System.out.println(l.getUserName()+l.getPassword()+user.getUserName()+use
r.getPassword());
		if(l.getUserName().equals(user.getUserName()) &&
l.getPassword().equals(user.getPassword()))
			return true;
		}
		catch(Exception e) {
			return false;
		}
		return false;
	}

}
```

STOCK :

```java
package org.model;

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

@Entity
public class Stock {
	@Id
	@GeneratedValue
	private int productId;
	private String productName;
	private int quantity;
	private float price;
	private String manufactureDate;
	public int getProductId() {
		return productId;
	}
	public String getProductName() {
		return productName;
	}
	public void setProductName(String productName) {
		this.productName = productName;
	}
	public int getQuantity() {
		return quantity;
	}
```

```
	public void setQuantity(int quantity) {
		this.quantity = quantity;
	}
	public float getPrice() {
		return price;
	}
	public void setPrice(float price) {
		this.price = price;
	}
	public String getManufactureDate() {
		return manufactureDate;
	}
	public void setManufactureDate(String manufactureDate) {
		this.manufactureDate = manufactureDate;
	}
	@Override
	public String toString() {
		return "\nproductId=" + productId + "\n productName=" + productName
+ "\n quantity=" + quantity + "\n price="
				+ price + "\n manufactureDate=" + manufactureDate;
	}

}
```

OUTPUT :

```
********* STOCK MANAGEMENT *********
USERNAME :
raama
PASSWORD :
123
Aug 27, 2017 7:16:08 PM org.hibernate.Version logVersion
INFO: HHH000412: Hibernate Core {5.2.8.Final}
Aug 27, 2017 7:16:08 PM org.hibernate.cfg.Environment <clinit>
INFO: HHH000206: hibernate.properties not found
Aug 27, 2017 7:16:08 PM org.hibernate.annotations.common.reflection.java.JavaReflectionManager <clinit>
INFO: HCANN000001: Hibernate Commons Annotations {5.0.1.Final}
Aug 27, 2017 7:16:08 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use!)
Aug 27, 2017 7:16:08 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001005: using driver [com.mysql.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/test]
Aug 27, 2017 7:16:08 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {user=root, password=****}
Aug 27, 2017 7:16:08 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
Aug 27, 2017 7:16:08 PM org.hibernate.engine.jdbc.connections.internal.PooledConnections <init>
INFO: HHH000115: Hibernate connection pool size: 10 (min=1)
Aug 27, 2017 7:16:09 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
Aug 27, 2017 7:16:09 PM org.hibernate.engine.jdbc.env.internal.LobCreatorBuilderImpl useContextualLobCreation
INFO: HHH000423: Disabling contextual LOB creation as JDBC driver reported JDBC version [3] less than 4
Aug 27, 2017 7:16:10 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess@7e3181aa] for
Hibernate: select login0_.userName as userName1_1_0_, login0_.password as password2_1_0_ from Login login0_ where login0_.userName=?
1.Add Product
2.View All
3.Update Product
4.Delete Product
5.Exit
2
```

```
Aug 27, 2017 7:16:12 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use!)
Aug 27, 2017 7:16:12 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001005: using driver [com.mysql.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/test]
Aug 27, 2017 7:16:12 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {user=root, password=****}
Aug 27, 2017 7:16:12 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
Aug 27, 2017 7:16:12 PM org.hibernate.engine.jdbc.connections.internal.PooledConnections <init>
INFO: HHH000115: Hibernate connection pool size: 10 (min=1)
Aug 27, 2017 7:16:12 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
Aug 27, 2017 7:16:12 PM org.hibernate.engine.jdbc.env.internal.LobCreatorBuilderImpl useContextualLobCreation
INFO: HHH000423: Disabling contextual LOB creation as JDBC driver reported JDBC version [3] less than 4
Aug 27, 2017 7:16:12 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess@6aecbb8d] for
Aug 27, 2017 7:16:12 PM org.hibernate.hql.internal.QueryTranslatorFactoryInitiator initiateService
INFO: HHH000397: Using ASTQueryTranslatorFactory
Hibernate: select stock0_.productId as productI1_2_, stock0_.manufactureDate as manufact2_2_, stock0_.price as price3_2_, stock0_.productName as productN4_2_, stock0_.quantity as qua

productId=2
 productName=soap
 quantity=2
 price=20.0
 manufactureDate=2-5-2017
1.Add Product
2.View All
3.Update Product
4.Delete Product
5.Exit
5
***BYE***
```