**EJB STATEFUL BEAN**

**Stateful Bean**

package com.hexa.service;

import javax.annotation.PostConstruct;

import javax.annotation.PreDestroy;

import javax.ejb.Stateful;

import javax.ejb.LocalBean;

import javax.ejb.PostActivate;

import javax.ejb.PrePassivate;

import javax.ejb.Remove;

@Stateful(name = "ibean1" , mappedName = "ibean2")

public class InterestBean implements InterestRemote{

private double amt;

private int years;

private float rate;

private String bname;

private static int i;

public InterestBean() {

bname = "hexa " + i;

++i;

System.out.println("Constructor Called");

}

@PrePassivate

public void mypassivate() {

System.out.println("mypassivate");

}

@PostActivate

public void myacive() {

System.out.println("myacive");

}

@PostConstruct

public void init(){

System.out.println("Post Construct");

}

@PreDestroy

public void preDestroy() {

System.out.println("Pre Destroy");

}

@Override

public void setData(double amt, int years, float rate) {

this.amt = amt;

this.years = years;

this.rate = rate;

}

@Override

public double calcSimple() {

return amt \* years \* rate / 100;

}

@Override

public double calcCompound() {

return amt \* Math.pow((1+rate/100), years) - amt;

}

@Override

@Remove

public void invalidate() {

System.out.println("Bean is yet to be removed");

}

@Override

public String getBname(){

return bname;

}

}}

**Interface**

package com.hexa.service;

import javax.ejb.Remote;

@Remote

public interface InterestRemote {

void setData(double amt, int years, float rate);

double calcSimple();

double calcCompound();

void invalidate();

String getBname();

}

Client Code:

package statefulclient;

import com.hexa.service.InterestRemote;

import java.util.Properties;

import javax.naming.Context;

import javax.naming.InitialContext;

import javax.naming.NamingException;

public class StatefulClient {

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

Properties props = new Properties();

props.setProperty(Context.INITIAL\_CONTEXT\_FACTORY, "com.sun.enterprise.naming.SerialInitContextFactory");

props.setProperty("com.omg.CORBA.ORBInitialHost", "localhost");

props.setProperty("com.omg.CORBA.ORBInitialPort", "3700");

Context ctx = new InitialContext(props);

InterestRemote ser = (InterestRemote) ctx.lookup("ibean2");

ser.setData(5000, 2, 5.5f);

System.out.println(ser.getBname());

System.out.println(ser.calcSimple());

System.out.println(ser.calcCompound());

//ser.invalidate();

}

}

**o/P:**

Constructor Called

Info: Post Construct

Info: Bean is yet to be removed

Info: Pre Destroy

**Dependency Injection**

package com.hexa.service;

import javax.ejb.Local;

@Local

public interface RoundLocal {

double round2Dec(double num);

}

package com.hexa.service;

import javax.ejb.Stateless;

import javax.ejb.LocalBean;

@Stateless(name = "round1" , mappedName = "round2")

public class RoundSession implements RoundRemote,RoundLocal {

@Override

public double round2Dec(double num) {

return (double) Math.round(num \* 100) / 100;

}

}

package com.hexa.service;

import javax.annotation.PostConstruct;

import javax.annotation.PreDestroy;

import javax.ejb.EJB;

import javax.ejb.Stateless;

import javax.ws.rs.container.PreMatching;

@Stateless(name = "interestbean1", mappedName = "interestbean2")

public class InterestService implements InterestSerRemote{

@EJB

private RoundLocal robj;

public InterestService() {

System.out.println("Session Bean - interest service - Constructor");

}

@PostConstruct

public void myinit() {

System.out.println("Post construct");

}

@PreDestroy

public void mydestroy() {

System.out.println("Pre destroy");

}

// Add business logic below. (Right-click in editor and choose

// "Insert Code > Add Business Method")

@Override

public double calcSimple(double amt,int years,float rate) {

double si = amt \* years \* rate / 100;

si = robj.round2Dec(si);

return si;

}

@Override

public double calcCompound(double amt, int years, float rate) {

double ci = amt \* Math.pow((1+rate/100), years) - amt;

ci = robj.round2Dec(ci);

return ci;

}

}

**Client Code:**

package statelessclient2;

import com.hexa.service.IHello;

import com.hexa.service.RoundLocal;

import com.hexa.service.RoundRemote;

import java.util.Properties;

import javax.naming.Context;

import javax.naming.InitialContext;

import javax.naming.NamingException;

public class RoundClient {

//

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

Properties props = new Properties();

props.setProperty(Context.INITIAL\_CONTEXT\_FACTORY, "com.sun.enterprise.naming.SerialInitContextFactory");

props.setProperty("org.omg.CORBA.ORBInitialHost", "localhost");

props.setProperty("org.omg.CORBA.ORBInitialPort", "3700");

Context ctx = new InitialContext(props);

RoundRemote ser = (RoundRemote) ctx.lookup("round2");

System.out.println(ser.round2Dec(76.587392));

}

}

**O/P:**

**Simple Interest390.0**

**Compound Interest399.83**

Message Driven Bean

JMS

1. JMS Sender Code
2. Client Code
3. MDB Code

**JMS SENDER CODE**

package com.hexa.service;

import javax.ejb.Remote;

@Remote

public interface JmsSendRemote {

public String sendMsg(String msg);

}

package com.hexa.service;

import javax.annotation.Resource;

import javax.ejb.Stateless;

import javax.ejb.LocalBean;

import javax.jms.Connection;

import javax.jms.ConnectionFactory;

import javax.jms.MessageProducer;

import javax.jms.Queue;

import javax.jms.Session;

import javax.jms.TextMessage;

@Stateless(name = "Jbean1" , mappedName = "Jbean2")

public class JmsSenderBean implements JmsSendRemote{

@Resource(mappedName = "java:comp/DefaultJMSConnectionFactory")

private ConnectionFactory cf;

@Resource(mappedName = "myqueue")

private Queue que;

@Override

public String sendMsg(String msg) {

try {

Connection con = cf.createConnection();

Session sess = con.createSession();

MessageProducer prod = sess.createProducer(que);

TextMessage tm = sess.createTextMessage(msg);

prod.send(tm);

sess.close();

con.close();

return "Your message is produced to Queue";

} catch(Exception e){

System.out.println(e.getMessage());

return "Your Message is not Produced";

}

}

}

**MDB CODE (Create Message driven Bean)**

package com.hexa.service;

import javax.ejb.ActivationConfigProperty;

import javax.ejb.MessageDriven;

import javax.jms.Message;

import javax.jms.MessageListener;

import javax.jms.TextMessage;

@MessageDriven(mappedName = "mbean1",activationConfig = {

@ActivationConfigProperty(propertyName = "destinationLookup", propertyValue = "myqueue"),

@ActivationConfigProperty(propertyName = "destinationType", propertyValue = "javax.jms.Queue")

})

public class MdbDemo1 implements MessageListener {

public MdbDemo1() {

}

@Override

public void onMessage(Message message) {

try{

TextMessage tm = (TextMessage) message;

System.out.println("Received From Queue : "+tm.getText());

} catch(Exception e) {

System.out.println(e.getMessage());

}

}

}

**Client Code:**

package statelessclient2;

import com.hexa.service.JmsSendRemote;

import com.hexa.service.RoundRemote;

import java.util.Properties;

import javax.naming.Context;

import javax.naming.InitialContext;

import javax.naming.NamingException;

public class JmsSendClient {

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

Properties props = new Properties();

props.setProperty(Context.INITIAL\_CONTEXT\_FACTORY, "com.sun.enterprise.naming.SerialInitContextFactory");

props.setProperty("org.omg.CORBA.ORBInitialHost", "localhost"); //172.25.207.118 localhost

props.setProperty("org.omg.CORBA.ORBInitialPort", "3700");

Context ctx = new InitialContext(props);

JmsSendRemote ser = (JmsSendRemote) ctx.lookup("Jbean2");

System.out.println(ser.sendMsg("Welcome Abhi"));

}

}

**O/P:**

Received From Queue : Welcome Abhi**Message Driven using Employee object**

**Employee.class**

package com.hexa.service;

import java.io.Serializable;

public class Emp implements Serializable{

private int eid;

private String ename;

public Emp(){

}

public int getEid() {

return eid;

}

public void setEid(int eid) {

this.eid = eid;

}

public String getEname() {

return ename;

}

public void setEname(String ename) {

this.ename = ename;

}

}

**Interface :**

package com.hexa.service;

import javax.ejb.Remote;

@Remote

public interface JmsSendRemote {

public String sendMsg(String msg);

public String sendMsg2(int eid,String ename);

}

**Sender Bean Class**

package com.hexa.service;

import javax.annotation.Resource;

import javax.ejb.Stateless;

import javax.ejb.LocalBean;

import javax.jms.Connection;

import javax.jms.ConnectionFactory;

import javax.jms.MessageProducer;

import javax.jms.ObjectMessage;

import javax.jms.Queue;

import javax.jms.Session;

import javax.jms.TextMessage;

@Stateless(name = "Jbean1" , mappedName = "Jbean2")

public class JmsSenderBean implements JmsSendRemote{

@Resource(mappedName = "java:comp/DefaultJMSConnectionFactory")

private ConnectionFactory cf;

@Resource(mappedName = "myqueue")

private Queue que;

@Override

public String sendMsg(String msg) {

try {

Connection con = cf.createConnection();

Session sess = con.createSession();

MessageProducer prod = sess.createProducer(que);

TextMessage tm = sess.createTextMessage(msg);

prod.send(tm);

sess.close();

con.close();

return "Your message is produced to Queue";

} catch(Exception e){

System.out.println(e.getMessage());

return "Your Message is not Produced";

}

}

@Override

public String sendMsg2(int eid, String ename) {

try{

Connection con = cf.createConnection();

Session sess = con.createSession();

MessageProducer prod = sess.createProducer(que);

Emp emp = new Emp();

emp.setEid(eid);

emp.setEname(ename);

ObjectMessage tm = sess.createObjectMessage(emp);

prod.send(tm);

sess.close();

con.close();

return "Your message is produced to Queue";

} catch(Exception e) {

System.out.println(e.getMessage());

return "Your Message is not Produced";

}

}

}

**MDB**

package com.hexa.service;

import javax.ejb.ActivationConfigProperty;

import javax.ejb.MessageDriven;

import javax.jms.Message;

import javax.jms.MessageListener;

import javax.jms.ObjectMessage;

import javax.jms.TextMessage;

@MessageDriven(mappedName = "mbean1",activationConfig = {

@ActivationConfigProperty(propertyName = "destinationLookup", propertyValue = "myqueue"),

@ActivationConfigProperty(propertyName = "destinationType", propertyValue = "javax.jms.Queue")

})

public class MdbDemo1 implements MessageListener {

public MdbDemo1() {

}

@Override

public void onMessage(Message message) {

try{

// The received message is later stored in DB for

if(message instanceof TextMessage){

TextMessage tm = (TextMessage) message;

System.out.println("Received From Queue : "+tm.getText());

} else {

ObjectMessage om = (ObjectMessage) message;

Emp emp = (Emp) om.getObject();

System.out.println("Received : " + emp.getEid() + " " + emp.getEname());

}

} catch(Exception e) {

System.out.println(e.getMessage());

}

}}

**Interceptor**

**MyInterceptor.java**

package com.hexa.service;

import javax.interceptor.AroundInvoke;

import javax.interceptor.InvocationContext;

public class MyInterceptor {

@AroundInvoke

public Object intercept(InvocationContext ctx) throws Exception{

String mtd = ctx.getMethod().getName();

String cls = ctx.getTarget().getClass().getSimpleName();

System.out.println("Logged : " + mtd + " of class " + cls);

Object[] arr = ctx.getParameters();

int pid = (int) arr[0];

if(pid >= 1001 && pid <=1003) {

Object obj = ctx.proceed();

System.out.println("Logged Out : " + mtd + "of class " + cls);

return obj;

}

return null;

}

}

**Product.java**

package com.hexa.service;

import java.io.Serializable;

public class Product implements Serializable{

private int pid;

private String pname;

private double price;

public Product(int pid, String pname, double price) {

this.pid = pid;

this.pname = pname;

this.price = price;

}

public int getPid() {

return pid;

}

public String getPname() {

return pname;

}

public double getPrice() {

return price;

}

}

**Product Remote Interface :**

package com.hexa.service;

import javax.ejb.Remote;

@Remote

public interface ProductRemote {

Product getProduct(int pid);

}

**Bean Class :**

package com.hexa.service;

import java.util.HashMap;

import java.util.Map;

import javax.ejb.Stateless;

import javax.ejb.LocalBean;

import javax.interceptor.Interceptors;

@Stateless(name = "pbean1", mappedName = "pbean2")

@Interceptors({MyInterceptor.class})

public class ProductService implements ProductRemote{

private static Map<Integer,Product> map;

static {

map = new HashMap<>();

map.put(1001, new Product(1001,"acer",23000));

map.put(1001, new Product(1002,"lenovo",53000));

map.put(1001, new Product(1003,"ibm",43000));

}

@Override

public Product getProduct(int pid) {

System.out.println("Inside Get Product");

return map.get(pid);

}

}

**Client Code :**

package statelessclient2;

import com.hexa.service.Product;

import com.hexa.service.ProductRemote;

import com.hexa.service.RoundRemote;

import java.util.Properties;

import javax.naming.Context;

import javax.naming.InitialContext;

import javax.naming.NamingException;

public class ProductClient {

//pbean2

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

Properties props = new Properties();

props.setProperty(Context.INITIAL\_CONTEXT\_FACTORY, "com.sun.enterprise.naming.SerialInitContextFactory");

props.setProperty("org.omg.CORBA.ORBInitialHost", "localhost");

props.setProperty("org.omg.CORBA.ORBInitialPort", "3700");

Context ctx = new InitialContext(props);

ProductRemote ser = (ProductRemote) ctx.lookup("pbean2");

Product prod = ser.getProduct(1009);

if(prod != null) {

System.out.println("Name : " + prod.getPname() + " Price : " + prod.getPrice());

} else {

System.out.println("Not Found");

}

}

}