```java
package controller;
import javax.ws.rs.*;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import java.util.List;
import model.Customer;
import service.CustomerService;

@Path("/customers")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
public class CustomerController {

    private CustomerService customerService = new CustomerService();

    @GET
    public Response getAllCustomers() {
        List<Customer> customers = customerService.getAllCustomers();
        return Response.ok(customers).build();
    }

    @GET
    @Path("/{id}")
    public Response getCustomerById(@PathParam("id") int id) {
        Customer customer = customerService.getCustomerById(id);
        if (customer != null) {
            return Response.ok(customer).build();
        } else {
            return Response.status(Response.Status.NOT_FOUND).entity("Customer not found").build();
        }
    }

    @POST
    public Response addCustomer(Customer customer) {
        customerService.addCustomer(customer);
        return Response.status(Response.Status.CREATED).entity("Customer added successfully").build();
    }

    @PUT
    @Path("/{id}")
    public Response updateCustomer(@PathParam("id") int id, Customer updatedCustomer) {
        Customer existingCustomer = customerService.getCustomerById(id);
        if (existingCustomer != null) {
            updatedCustomer.setCusId(id);
            customerService.updateCustomer(updatedCustomer);
            return Response.ok("Customer updated successfully").build();
        } else {
            return Response.status(Response.Status.NOT_FOUND).entity("Customer not found").build();
        }
    }

    @DELETE
    @Path("/{id}")
    public Response deleteCustomer(@PathParam("id") int id) {
        customerService.deleteCustomer(id);
        return Response.ok("Customer deleted successfully").build();
    }

    @GET
    @Path("/search")
    public Response searchCustomers(@QueryParam("query") String query) {
        if (query == null || query.trim().isEmpty()) {
            return Response.status(Response.Status.BAD_REQUEST)
                    .entity("Query parameter is required").build();
        }
        List<Customer> customers = customerService.searchCustomers(query);
        return Response.ok(customers).build();
    }
} http://localhost:8080/b/rest/customers/search?query=aom
```

```java
package service;
import java.util.List;
import org.hibernate.Session;
import org.hibernate.Transaction;
import org.hibernate.query.Query;
import model.Customer;
import util.SessionUtil;
public class CustomerService {
    public void addCustomer(Customer customer) {
        Transaction transaction = null;
        try (Session session = SessionUtil.getSession()) {
            transaction = session.beginTransaction();
            session.save(customer);
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) transaction.rollback();
            e.printStackTrace();
        }
    }
    public List<Customer> getAllCustomers() {
        try (Session session = SessionUtil.getSession()) {
            Query<Customer> query = session.createQuery("FROM Customer", Customer.class);
            return query.list();
        }
    }

    public Customer getCustomerById(int id) {
        try (Session session = SessionUtil.getSession()) {
            return session.get(Customer.class, id);
        }
    }

    public void updateCustomer(Customer customer) {
        Transaction transaction = null;
        try (Session session = SessionUtil.getSession()) {
            transaction = session.beginTransaction();
            session.update(customer);
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) transaction.rollback();
            e.printStackTrace();
        }
    }

    public void deleteCustomer(int id) {
        Transaction transaction = null;
        try (Session session = SessionUtil.getSession()) {
            transaction = session.beginTransaction();
            Customer customer = session.get(Customer.class, id);
            if (customer != null) {
                session.delete(customer);
            }
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) transaction.rollback();
            e.printStackTrace();
        }
    }
    public List<Customer> searchCustomers(String keyword) {
        try (Session session = SessionUtil.getSession()) {
            Query<Customer> query = session.createQuery(
                "FROM Customer c WHERE LOWER(c.name) LIKE :keyword OR LOWER(c.username) LIKE :keyword",
                Customer.class
            );
            query.setParameter("keyword", "%" + keyword.toLowerCase() + "%");
            return query.list();
        }
    }
}
```

```xml
<?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>
        <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/soa</property>
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
        <property name="hibernate.connection.username">root</property>
        <mapping resource="Customer.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```