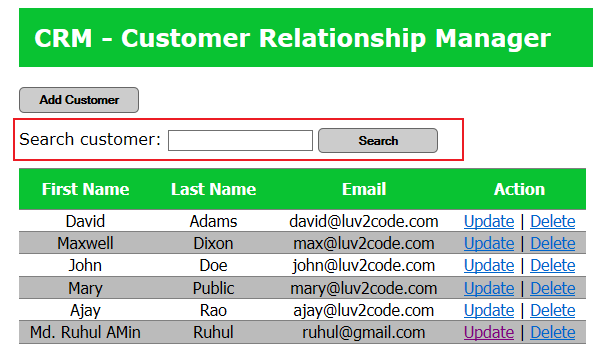
**33.1. Search Customer**

**Add Search feature in our app**:

Now we will create a search option in our application. We will allow the user to search for a customer by name. We'll add a search box to the screen and the user can enter a name. On the backend, we'll compare this name to the customer's first name or last name.



**Developing Process (Step-by-Step)**:

1. Create the HTML form
2. Add mapping to the controller
3. Add methods in the service layer to delegate to DAO
4. Add method in the DAO to perform search

**1) Create HTML form**:

Add the following code bellow the "**Add Customer**" button.

<!-- add a search box -->

<form:form action=*"search"* method=*"GET"*>

Search customer: <input type=*"text"* name=*"theSearchName"* />

<input type=*"submit"* value=*"Search"* class=*"add-button"* />

</form:form>

**2) Add mapping to the controller**:

You need to add a mapping to handle the search form submission

1. Edit the file: CustomerController.java
2. Add the new mapping and method

// mapping for theSearchName

@GetMapping("/search")

**public** String searchCustomers(@RequestParam("theSearchName")String theSearchName,

Model theModel){

// search customer from the service

List<Customer> theCustomers = customerService.searchCustomers(theSearchName);

// add the customer to the model

theModel.addAttribute("customers", theCustomers);

**return** "list-customer";

}

**Note**:

Here we can get GET or POST method. If we use GET in form, then we have to use @GetMapping in Controller class. And if we use POST in form, then we have to use @PostMapping in Controller class.

You may have syntax errors on the customerService, but we'll resolve that in the next section.

**3) Add methods in the service layer to delegate to DAO**:

You need to add methods in the service layer to delegate calls to the DAO

a) Edit the file: CustomerService.java and add the method declaration

**public** List<Customer> searchCustomers(String theSearchName);

b) Edit the file: CustomerServiceImpl.java and add the method

@Override

@Transactional

**public** List<Customer> searchCustomers(String theSearchName) {

**return** customerDAO.searchCustomers(theSearchName);

}

You may have syntax errors on the customerDAO, but we'll resolve that in the next section.

**4) Add method in the DAO to perform search**:

Now, we'll add methods in the DAO layer to search for a customer by first name or last name.

a) Edit the file: CustomerDAO.java and add the method declaration

**public** List<Customer> searchCustomers(String theSearchName);

b) Edit the file: CustomerDAOImpl.java and add the method:

@Override

**public** List<Customer> searchCustomers(String theSearchName) {

// get the current Hbernate session

Session currentSession = sessionFactory.getCurrentSession();

Query<Customer> theQuery;

// only search by name if theSearchName is not empty

**if** (theSearchName != **null** && theSearchName.trim().length() > 0) {

// search for firstName or lastName ... case insensitive

theQuery = currentSession.createQuery("from Customer wherelower(firstName) like :theName or lower(lastName) like :theName",Customer.**class**);

theQuery.setParameter("theName", "%" +theSearchName.toLowerCase() +

"%");

}

**else** {

// theSearchName is empty ... so just get all customers

theQuery = currentSession.createQuery("from Customer", Customer.**class**);

}

// execute query and get the result

List<Customer> customers = theQuery.getResultList();

// return the result

**return** customers;

}

In this method, we need to check "**theSearchName**", this is the user input. We need to make sure it is not empty. If it is not empty, then we will use it in the search query. If it is empty, then we'll just ignore it and simply return all of the customers.

For the condition when "**theSearchName**" is not empty, then we use it to compare against the first name or last name. We also make use of the "like" clause and the "%" wildcard characters. This will allow us to search for substrings. For example, if we have customers with last name of "Patel", "Patterson" ... then we can search for "Pat" and it will match on those names.

Also, notice the query uses the lower case version of the values to make a case insensitive search. If you'd like to make a case sensitive search, then simply remove the lower references.

33.1. Search Customer