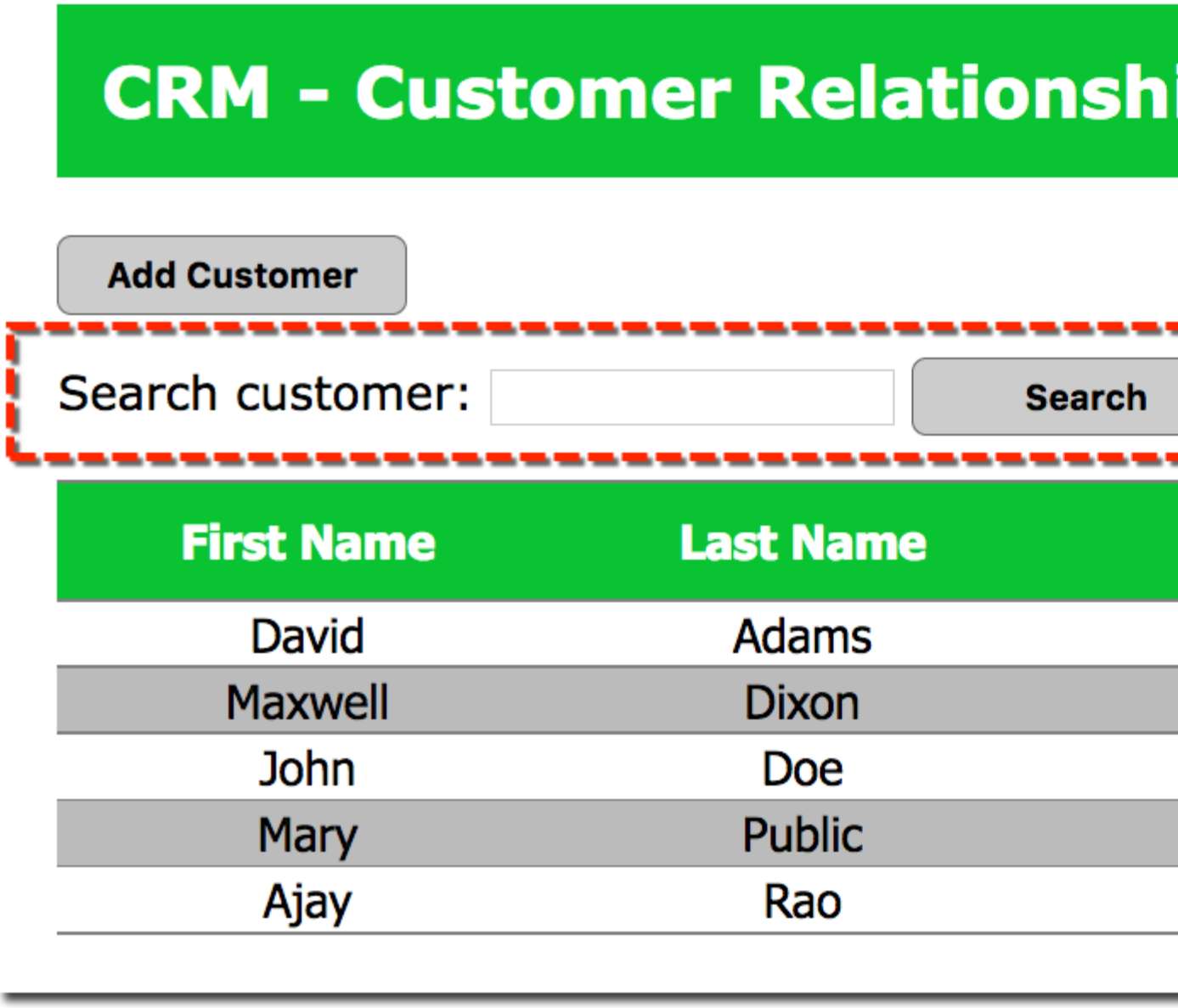
FAQ: How to Add Search support

FAQ: How to add Search features to the App?

A number of students have asked how to add search features to the app? I'll walk through this process.

Basically, we will allow the user to search for a customer by name. We'll add a search box at 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.



Overview of Development Process

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 perfom search

Download Source Code

All of the complete solution code is available for download here.

http://www.luv2code.com/downloads/udemy-spring-hibernate/crm-tracker-bonus-search.zip

I show you the detailed steps below so you can see what is added to each file.

Detailed Steps

1. Create the HTML form

You need to add a search form to read the user input and submit it to your Spring controller mapping

a. Edit the file: list-customers.jsp

b. We'll need to use Spring FORM tags, so at the top of the file, add the following taglib reference

```
<%@ taglib prefix="form"
uri="http://www.springframework.org/tags/form" %>
```

c. Now add a search form right after the search 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

a. Edit the file: CustomerController.java

b. Add the new mapping and method

```
    @GetMapping("/search")
    public String searchCustomers(@RequestParam("theSearchName")
String theSearchName,
                                  Model theModel) {

        // search customers from the service
        List<Customer> theCustomers =
customerService.searchCustomers(theSearchName);

        // add the customers to the model
        theModel.addAttribute("customers", theCustomers);

        return "list-customers";
    }
```

c. 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

b. Add the method declaration

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

c. Edit the file: CustomerServiceImpl.java

d. Add the method:

```
    @Override
    @Transactional
    public List<Customer> searchCustomers(String theSearchName) {
```

```
        return customerDAO.searchCustomers(theSearchName);
    }
```

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

---

4. Add method in the DAO to perfom 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

b. Add the method declaration

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

c. Edit the file: CustomerDAOImpl.java

d. Add the method:

```
    @Override
    public List<Customer> searchCustomers(String theSearchName) {

        // get the current hibernate session
        Session currentSession = sessionFactory.getCurrentSession();

        Query theQuery = null;

        //
        // 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 where
lower(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 result list
        List<Customer> customers = theQuery.getResultList();

        // return the results
        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.

You can read more on the HQL "like" clause here:
http://docs.jboss.org/hibernate/orm/5.2/userguide/html_single/Hibernate_User_Guide.html#hql-like-predicate

---

5. Test the app

Once you've made all of the updates then you can test your application.

The app will now have the search form at the top. You can enter a name to search and they app will give you the desired results.

Congrats!