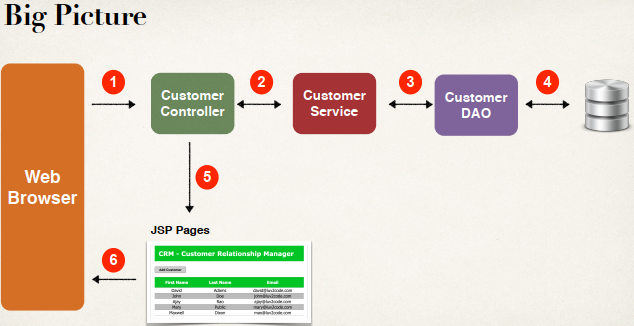
**Add New Customer (Step-by-Step)**

**Development Process**:



**Add a new Customer (Step-by-Step)**:

To add a new customer in our database we have to follow the following steps.

1. Update "**list-customer.jsp**"
   1. New "**Add Customer**" button
2. Create HTML form ("**customer-form.jsp**") for new customer
3. Process Form Data
   1. Controller -> Service -> DAO

**1) Update list-customer.jsp**:

<div id=*"container"*>

<div id=*"content"*>

<!-- put new button: Add Customer -->

<!—Return false cancels the "default" behavior in JavaScript -->

<input type=*"button"* value=*"Add Customer"*

onclick="window.location.href='showFormForAdd'; return false"

class=*"add-button"* />

<!-- Add out html table here -->

<table>

...

</table>

</div>

</div>

After run our project we will see the add button in "**list-customer**" page. But if we click the button, we will get 404 error. Because we don’t setup the link "**showFormForAdd**".

We have to add a request mapping in our controller for "**showFormForAdd**" to show a from.

**2) Create HTML form ("customer-form.jsp") for new customer**:

**File: customer-form.jsp**:

<!DOCTYPE html>

<html>

<head>

<title>Customer Form</title>

</head>

<body>

<h3>Customer Form</h3>

</body>

</html>

Now we add mapping for "**showFormForAdd**" in "**CustomerController**".

**File: CustomerController**:

// display customer-form

@GetMapping("/showFormForAdd")

**public** String showFormForAdd(Model theModel) {

**return** "customer-form";

}

**File: customer-form.jsp (Complete)**:

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

<!DOCTYPE html>

<html>

<head>

<title>Save Customer</title>

<link type=*"text/css"*

rel=*"stylesheet"*

href=*"*${pageContext.request.contextPath}*/resources/css/style.css"* />

<link type=*"text/css"*

rel=*"stylesheet"*

href=*"*${pageContext.request.contextPath}*/resources/css/add-customer-style.css"* />

</head>

<body>

<div id=*"wrapper"*>

<div id=*"header"*>

<h2>CRM - Customer Relationship Manager</h2>

</div>

</div>

<div id=*"container"*>

<h3>Save Customer</h3>

<form:form action=*"saveCustomer"* modelAttribute=*"customer"* method=*"POST"*>

<table>

<tbody>

<tr>

<td><label>First Name:</label></td>

<td><form:input path=*"firstName"*/></td>

</tr>

<tr>

<td><label>Last Name:</label></td>

<td><form:input path=*"lastName"*/></td>

</tr>

<tr>

<td><label>Email:</label></td>

<td><form:input path=*"firstName"*/></td>

</tr>

<tr>

<td><label></label></td>

<td><input type=*"submit"* value=*"Save"* class=*"save"*/></td>

</tr>

</tbody>

</table>

</form:form>

<!-- Add a Navigation Link -->

<div style="clear; both"></div>

<p>

<a href=*"*${pageContext.request.contextPath}*/customer/list"*>Back to List</a>

</p>

</div>

</body>

</html>

**CustomerController**:

// display customer-form (Add Customer)

@GetMapping("/showFormForAdd")

**public** String showFormForAdd(Model theModel) {

// create model attribute to bind form data

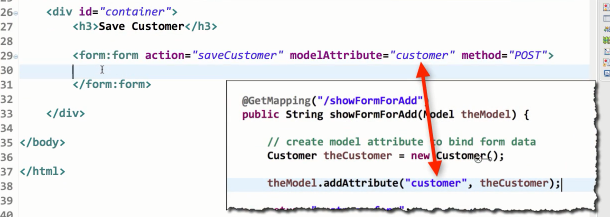
Customer theCustomer = **new** Customer();

theModel.addAttribute("customer", theCustomer);

**return** "customer-form";

}

**Diagram:**

****

**1) Process Form Data**:

Controller -> Service -> DAO

**CustomerController**:

// save customer -> customer-form (Add Button)

@PostMapping("/saveCustomer")

**public** String saveCustomer(@ModelAttribute("customer") Customer theCustomer) {

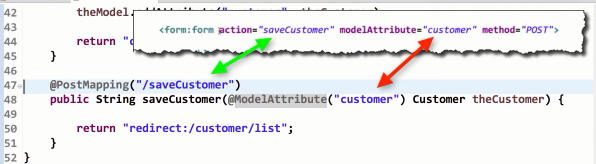
// save the customer using service

customerService.saveCustomer(theCustomer);

**return** "redirect:/customer/list";

}

**Diagram**:



**Update Service:**

**CustomerService:**

**public** **interface** CustomerService {

**public** List<Customer> getCustomers();

**public** **void** saveCustomer(Customer theCustomer);

}

**CustomerServiceImpl:**

@Service

**public** **class** CustomerServiceImpl **implements** CustomerService {

//need to inject CustomerDAO

@Autowired

**private** CustomerDAO customerDAO;

@Override

@Transactional

**public** List<Customer> getCustomers() {

**return** customerDAO.getCustomers();

}

@Override

@Transactional

**public** **void** saveCustomer(Customer theCustomer) {

customerDAO.saveCustomer(theCustomer);

}

}

**Update DAO**:

**CustomerDAO:**

**public** **interface** CustomerDAO {

**public** List<Customer> getCustomers();

**public** **void** saveCustomer(Customer theCustomer);

}

**CustomerDAOImp:**

@Repository

**public** **class** CustomerDAOImp **implements** CustomerDAO {

// need to inject the session factory

@Autowired

SessionFactory sessionFactory;

@Override

**public** List<Customer> getCustomers() {

// get the current hibernate session

Session currentSession = sessionFactory.getCurrentSession();

// create a query

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

// execute query and get result list

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

// return the result

**return** customers;

}

@Override

**public** **void** saveCustomer(Customer theCustomer) {

// get current Hibernate session

Session currentSession = sessionFactory.getCurrentSession();

// save the customer ...

currentSession.save(theCustomer);

}

}

**Sort Customer Data**:

For sort Customer data we have to update our Query in **CustomerDAOImp** class.

Query<Customer> theQuery = currentSession.createQuery("from Customer order by lastName", Customer.**class**);

**CustomerDAOImp.java**

@Repository

**public** **class** CustomerDAOImp **implements** CustomerDAO {

// need to inject the session factory

@Autowired

SessionFactory sessionFactory;

@Override

**public** List<Customer> getCustomers() {

// get the current hibernate session

Session currentSession = sessionFactory.getCurrentSession();

// create a query

Query<Customer> theQuery = currentSession.createQuery("from Customer order by lastName", Customer.**class**);

// execute query and get result list

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

// return the result

**return** customers;

}

@Override

**public** **void** saveCustomer(Customer theCustomer) {

// get current Hibernate session

Session currentSession = sessionFactory.getCurrentSession();

// save the customer ...

currentSession.save(theCustomer);

}

}

Controller -> Service -> DAO