Spring MVC Tutorial – Contact List Application

Step 09: Wiring the View to CRUD REST Controller Endpoints

Step 09: Wiring the View to CRUD REST Controller Endpoints

Overview

In the previous steps, we created our in-memory DAO, designed and implemented the REST endpoints for CRUD functionality in the controller, and wired dummy data to our Home page. In this step, we'll add JavaScript code to the Home view component that will allow us to request live data from the server (via Ajax) and display it.

What is Ajax?

Ajax (also AJAX) is short for **Asynchronous JavaScript and XML**. Ajax uses a combination of JavaScript, CSS, HTML, and either XML/XSLT or JSON to allow client/view components of a web application to send and receive data from the server asynchronously. Although XML is in the name, XML is not required for data exchange — in fact, JSON is a more popular choice in modern applications. When the client/view component receives data from the server, the user interface is updated dynamically using the DOM and JavaScript.

Creating a New Contact

In our application, new Contacts are created when the New Contact form is filled out and the Create Contact button is clicked. Although this form looks like a normal HTML form, it is not. We do not want to send data to the server by submitting the form and having the entire page refresh. Instead, we want to detect the button click, bundle the data from the form into a JSON object, and then send an Ajax request to our Create Contact endpoint. If the call is successful, we then want to clear the data from the form and reload the Contact Summary table so that the newly-created Contact is displayed. Add the following code to the Document Ready section of your contactList.js file:

```
// Document ready function
$(document).ready(function () {
    loadContacts();

    // NEW CODE START
    // on click for our add button
    $('#add-button').click(function (event) {
        // we don't want the button to actually submit
        // we'll handle data submission via ajax
        event.preventDefault();
        // Make an Ajax call to the server. HTTP verb = POST, URL = contact
        $.ajax({
            type: 'POST',
            url: 'contact',
            // Build a JSON object from the data in the form
            data: JSON.stringify({
                firstName: $('#add-first-name').val(),
                lastName: $('#add-last-name').val(),
                company: $('#add-company').val(),
                phone: $('#add-phone').val(),
                email: $('#add-email').val()
            }),
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            'dataType': 'json'
        }).success(function (data, status) {
            // If the call succeeds, clear the form and reload the summary table
            $('#add-first-name').val('');
            $('#add-last-name').val('');
            $('#add-company').val('');
            $('#add-phone').val('');
            $('#add-email').val('');
            loadContacts();
            //return false;
        });
    });
```

Getting and Displaying All Contacts

In a previous step, we created a loadContacts function in contactList.js. The initial version of this function dynamically displayed a hard-coded array of JSON test objects. We will now add code to loadContacts so that it will ask the server for all of the Contacts in the application and then render this data from the server into the Contact Summary table. Modify your loadContacts function so it looks like the following (note that the code below will register an onClick handler for the Delete link; we will implement this handler later in this document):

```
// Load contacts into the summary table
function loadContacts() {
    // clear the previous list
    clearContactTable();
    // grab the tbody element that will hold the new list of contacts
    var cTable = $('#contentRows');
    // Make an Ajax GET call to the 'contacts' endpoint. Iterate through
    // each of the JSON objects that are returned and render them to the
    // summary table.
    $.ajax({
        url: "contacts"
    }).success(function (data, status) {
        $.each(data, function (index, contact) {
            cTable.append($('<tr>')
                    .append($('<td>')
                            .append($('<a>')
                                    .attr({
                                        'data-contact-id': contact.contactId,
                                        'data-toggle': 'modal',
                                        'data-target': '#detailsModal'
                                    })
                                    .text(contact.firstName + ' ' + contact.lastName)
                                    ) // ends the <a> tag
                            ) // ends the <td> tag for the contact name
                    .append($('<td>').text(contact.company))
                    .append($('<td>')
                            .append($('<a>')
                                    .attr({
                                        'data-contact-id': contact.contactId,
                                        'data-toggle': 'modal',
                                        'data-target': '#editModal'
                                    })
                                    .text('Edit')
                                    ) // ends the <a> tag
                            ) // ends the <td> tag for Edit
                    .append($('<td>')
                            .append($('<a>')
                                    .attr({
                                        'onClick': 'deleteContact(' + contact.contactId + ')'
                                    })
                                    .text('Delete')
                                    ) // ends the <a> tag
                            ) // ends the <td> tag for Delete
                    );  // ends the <tr> for this Contact
        });  // ends the 'each' function
    });
}
```

Getting a Contact and Displaying Details

We will also add some JavaScript code that will fill the Details modal with live data rather than the canned test data we have been using so far. This requires a change to the Details modal show.bs.modal event handler — we replace the code that loads the canned test data into the modal with code that makes an Ajax call and then fills the modal with the JSON data returned by the server. Change your Details modal show.ms.modal event handler so it looks like the following:

```
// This code runs in response to show.bs.modal event for the details Modal
$('#detailsModal').on('show.bs.modal', function (event) {
    // get the element that triggered the event
    var element = $(event.relatedTarget);
    var contactId = element.data('contact-id');

    var modal = $(this);

    // make an ajax call to get contact information for given contact id
    // this is a GET request to contact/{id}
    // upon success, put the returned JSON data into the modal dialog
    $.ajax({
        type: 'GET',
        url: 'contact/' + contactId
    }).success(function (contact) {
        modal.find('#contact-id').text(contact.contactId);
        modal.find('#contact-firstName').text(contact.firstName);
        modal.find('#contact-lastName').text(contact.lastName);
        modal.find('#contact-company').text(contact.company);
        modal.find('#contact-phone').text(contact.phone);
        modal.find('#contact-email').text(contact.email);
    });
});
```

Getting a Contact and Filling Edit Form

Now, we'll change the Edit modal show.bs.modal much like we changed the Details modal above. Change your Details modal show.bs.modal event handler so it looks like the following:

```
// This code runs in response to the show.hs.modal event for the edit Modal
$('#editModal').on('show.bs.modal', function (event) {
    var element = $(event.relatedTarget);
    var contactId = element.data('contact-id');

    var modal = $(this);

    $.ajax({
        type: 'GET',
        url: 'contact/' + contactId
    }).success(function (contact) {
        modal.find('#contact-id').text(contact.contactId);
        modal.find('#edit-contact-id').val(contact.contactId);
        modal.find('#edit-first-name').val(contact.firstName);
        modal.find('#edit-last-name').val(contact.lastName);
        modal.find('#edit-company').val(contact.company);
        modal.find('#edit-email').val(contact.email);
        modal.find('#edit-phone').val(contact.phone);
    });
});
```

Editing a Contact

Our next step is to wire up the Edit button's click event to JavaScript code that will gather the Edit form data into a JSON object and send an Ajax PUT request to the server to update the Contact. To do this, we must create an onClick handler for Edit button, similar to the one we created for the Add button. Add the following code just after the Add button click handler code:

```
// onclick handler for edit button
$('#edit-button').click(function (event) {
    // prevent the button press from submitting the whole page
    event.preventDefault();

    // Ajax call -
    // Method - PUT
    // URL - contact/{id}
    // Just reload all of the Contacts upon success
    $.ajax({
        type: 'PUT',
        url: 'contact/' + $('#edit-contact-id').val(),
        data: JSON.stringify({
            contactId: $('#edit-contact-id').val(),
            firstName: $('#edit-first-name').val(),
            lastName: $('#edit-last-name').val(),
            company: $('#edit-company').val(),
            phone: $('#edit-phone').val(),
            email: $('#edit-email').val()
        }),
        headers: {
            'Accept': 'application/json',
            'Content-Type': 'application/json'
        },
        'dataType': 'json'
    }).success(function () {
        loadContacts();
    });
});
```

Deleting a Contact

Our final step here is to add delete functionality to our application. The code that we entered in the Getting and Displaying All Contacts section included the registration of an onClick handler for the Delete anchor. We will now implement that function. Add the following code just after the loadContacts function:

```javascript
function deleteContact(id) {
    var answer = confirm("Do you really want to delete this contact?");

    if (answer === true) {
        $.ajax({
            type: 'DELETE',
            url: 'contact/' + id
        }).success(function () {
            loadContacts();
        });
    }
}
```