CSCI-201: Principles of Software Development

Spring 2022

Lab6: HTML Form Validation

Introduction:

Accepting form submissions is one of the primary ways of getting information from users through a web page. However, since a user can type anything at all into form fields, we need to validate the data to ensure accuracy.

Implementation:

Start with creating a *Dynamic Web Project* in Eclipse. Remember that all the frontend files (HTML) go into the *src/main/webapp* directory. Don't worry about including CSS in your page unless you have time and want to make it look a little nicer.

In your HTML file you need to display a number of different form fields. The content of the form can be whatever you want (user registration, items to purchase, types and descriptions of animals, etc.).

text	textarea	select box
radio buttons	month	email
range	submit	reset

Make sure you uniquely identify each input by setting up their name attributes.

Although there may be some validation that is occurring through HTML5, you will need to submit all those fields to a servlet. You can specify the address (i.e. URL) of the servlet through the action attribute in the form tag. Note that you will also need to specify the HTTP method through the method attribute. We will be using the GET method for this lab.

Now, it is the time for you to create the servlet in the backend. Please note, in a Dynamic Web Project, all the backend files go into the *src/main/java* directory under Java Resources. Note that the @WebServlet() annotation defines the URL of the servlet. Make sure the action attribute of the form has that URL.

You can retrieve the form fields in the servlet through *HttpServletRequest* (i.e. the request variable). Here's a method that you may find helpful: *request.getParameter().*

The servlet will then validate that the format of each field is correct AND that every field was required. If any field was not correct or was blank, redirect the page back to the form using the *RequestDispatcher* interface. You should print a descriptive error message next to each of the fields that are incorrect.

If all the data was submitted properly, display it all in a table for the user to see. This should be done by redirecting from the servlet to a JSP. Make use of the *RequestDispatcher* for redirecting. See lecture 12 for redirecting coding example. The *forward* method of the *RequestDispatcher* forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.

Now, go back to your JSP. You will need to add some Java code to make sure that if the servlet sends the user back to the form page, the previous values will be reloaded to their respective fields. Remember that the *forward* method in *RequestDispatcher* also forwards the *HttpServletRequest*, which is where all the field values are stored. Be sure to display the error messages as well.

Finally, create a second JSP to display the user inputs in a table.

In short, you should have three different files for this lab – the initial form page, the servlet for validation, and the JSP if successful.

Grading Criteria:

Labs are graded based on your understanding of the course material. To receive full credit, you will need to 1) complete the lab following the instructions above **AND** 2) show your understanding of the lab material by answering questions upon check-off.

Check-off Questions:

Instructors, please randomly select one question from each section.

Question 1

a) Out of the three files that we just created, which files are on the frontend and which are on the backend? Please explain how the frontend differs from the backend.

b) Out of the three files that we just created, which files are on the server side and which are on the client side? What are the types of file that typically run on the server side? What about client side?

c) In this lab, is the form validation done on the server or the client? Please explain.

d) Draw a flow chart to illustrate the relationship among the three files.

Question 2

a) With a tiny change, we can connect the initial form page directly to the results page without going through the servlet. What is the change that we can make?

b) Why do we need to specify the HTTP method? What is the other method typically used for form submission and what is the difference between them?

c) What happens if we don't have the *name* attribute for input fields?

Question 3

a) In the implementation above, where do we generate the error messages for invalid inputs?

b) How do we specify the desired servlet when validating forms?

c) When calling the *forward* method in *RequestDispatcher*, what is the one extra thing that we will need to do to make the code compile? Why is that generally not needed for other methods?