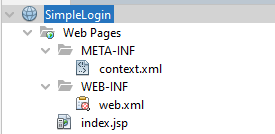
<https://www.javatpoint.com/requestdispatcher-in-servlet>

<https://initialcommit.com/blog/how-to-send-data-from-servlet-to-jsp>

So you really only need Three files

The first file you should make is the JSP. This will go into whatever folder you set. I just left it in the web pages folder in the hierarchy. YOu can make it with the +new file option then clicking jsp



You make the form in html inside your JSP

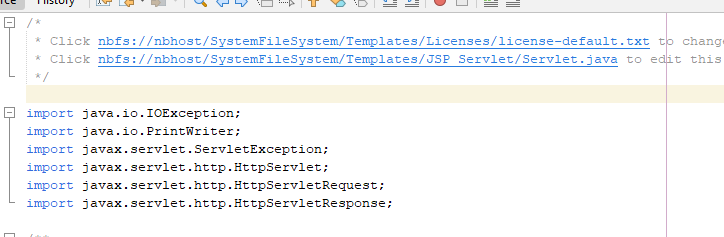


Make sure the form action matches the URL you will set in the web.xml file

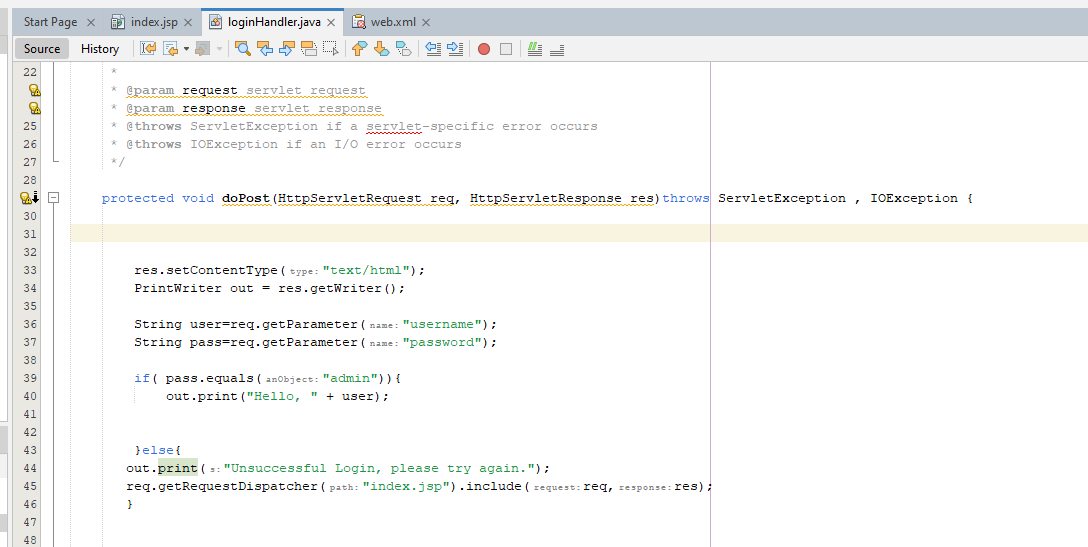
Now we will make the servlet. I made the servlet by using the +new option and selecting servlet.

I named mine loginHandler

The following imports will be in the servlet that is auto generated by netbeans



I then deleted all the other methods just for the sake of making my code simpler BUT you can replace your do post function with min



First I declared the content type, I am not sure what it does but I assume its just saying that whatever response you're going to send is html/text. Then I created a PrintWriter object. This allows me to write html text into the jsp from the servlet. Think of it as a print statement

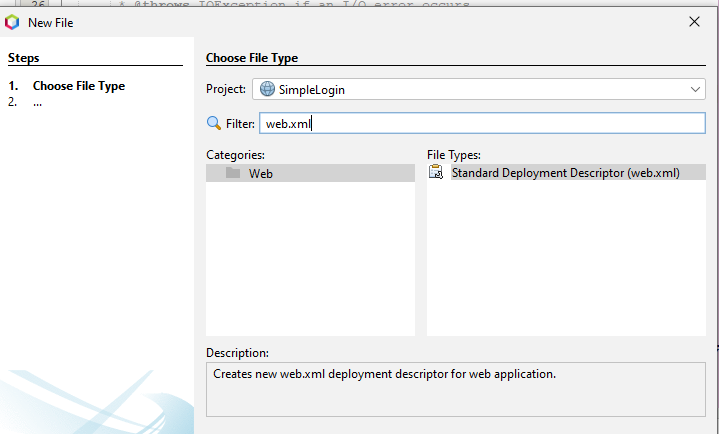
Then I stored the username and password in some variables. I got the parameters from the jsp form using the get parameter function. Make sure your function arguments match the name (name = “YOUR NAME HERE” in the jsp form tags) given in the jsp form. It then checks to make sure if the password is “admin”.

If it is true it runs an if statement that prints text. This is like a write option where it erases all the html on the page associated with the servlet, in this case our log in page, and replaces it with our printed data. I made it print hello and then the persons name as like a pseudo home page.

Else it prints the unsuccessful login message BUT this time i followed it up with that long request function. It pretty much boils down to the first set of parentheses is the page you're trying to put our out print function on then the second set of parenthesis is for youre request and response object. An important note is you can put .forward instead of .include if you want to send info to a different page. I used .include since I wanted to just update the page, I am not sure if this is correct fully but from my understanding the .include function is like append information to a page and .forward is to send a message to a page and only that new message is displayed or in some cases its handed to another servlet and it gets handled from there. In this example <https://www.javatpoint.com/requestdispatcher-in-servlet> the .forward is used to have another servlet generate its own landing page .

Finally the most important part the xml file

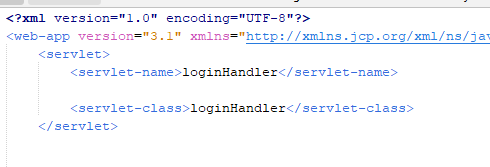
The web.xml can be made by looking it up specifically by “web.xml” in the new file option



This is how we actually make the links for pages and connect servlets and jsps.

It should automatically be created under WEB-INF but if not I'd drag it there because that's how mine was created by netbeans.

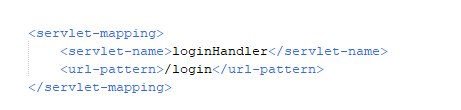
The first part is actually creating our servlet object inside of web.xml. The first thing we do is between two servlet tags we define the servlet name and class



From what I understand the name is just how you refer to it inside the code and the class is the actual java file you created earlier that is a servlet. I don't know if the names have to match but I normally see them that way in the tutorial and on stack overflow so I’d write it that way.

Side Note (**This doesn't need to be typed into your code**): I believe we can associate specific JSPS with a servlet in this part of the process I saw once on a stack overflow post (<https://stackoverflow.com/questions/6401588/servlet-jsp-web-xml>) that you can associate a jsp to a servlet by adding it with the jsp-file tag in between the servlet tags but I am not too sure. I decided to note this since we may need this in the future because the post talked about using it for precompiling jsps that have onload functions with them. Regardless I just felt like this might be important later.

Now we have to actually map the servlet to a url. We do this with two servlet-mapping tags and then using a servlet name tag and url pattern tag in between.



The servlet name is the same one you want to refer to inside the servlet definition. In this case we want all the stuff that's named loginHandler to be given the sub url of /login.if we used /login in the url pattern then the full url would be https://example.com/Login .

Finally you can set your default starting page with the welcome file list tags like so.

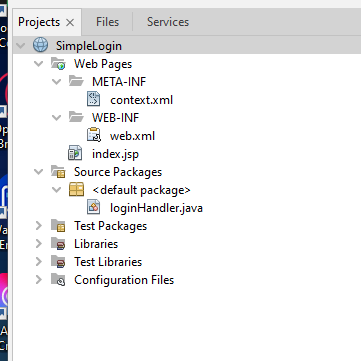


Overall your web.xml should look like this



I did have some extra code in here when I let netbeans generate my webn.xml. The only one I left in there during my final run where everything worked was the session tags but I deleted mine so that way my code only has the exact things I need.

To launch the software just right click the project name in the file tree (the glove icon file at the top) and press run. It should automatically open the webpage in a browser.



If you have issues during development just right click and press run again and it'll relaunch the code. You will need to relaunch everytime a change is made.

