**CSCI 201 Lab 5**

**JSP, Servlets, JavaScript**

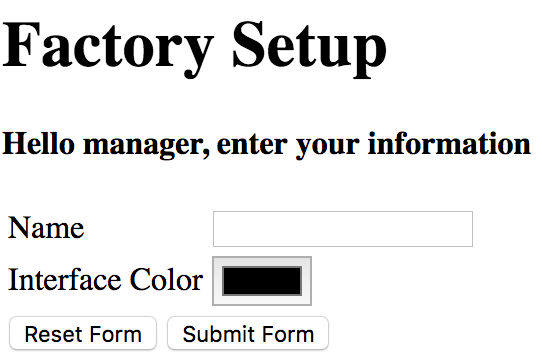
*People think that computer science is the art of geniuses but the actual reality is the opposite, just many people doing things that build on each other, like a wall of mini stones. – Donald Knuth*

You, as a user of this factory simulation, are already a spectator of sorts, so let’s make you a manager. You will have some custom info rendered in the user interface and track the workers’ performance to be sent to a summary page.

**Part 1 – Setup Form**

Create an HTML called setupForm.html.

Add a form to take in your name and favorite color.



**Notes:**

Your form’s action should forward to “../SetupServlet” (or whichever path leads you to SetupServlet), which we will write next. We will use its doGet(…) method.

../../../../Screen%20Shot%202017-02-07%20at%206.44.07%20AM.png

Remember the value you give the “name” attribute, this is how you will access the form information in your servlet.

For example, here we used “color”

../Screen%20Shot%202017-02-05%20at%207.08.42%20PM.png

So in our servlet we will call

../Screen%20Shot%202017-02-05%20at%207.06.06%20PM.png

If you get stuck, use schoolform.html as a reference.

**Part 2 – Passing Data**

We are following the Model, View, Controller (MVC) design pattern, so our View (in this case, setupForm.html), will send our Controller (a servlet) our name and color to be stored in a Model (an HttpSession). You can find this pattern throughout the factory code; as one example:

Model: Factory.java (from factory.txt)

View: index.html

Controller: Factory.js

Let’s make a SetupServlet.java.

Get the user’s name parameter from the request:

../Screen%20Shot%202017-02-05%20at%207.06.29%20PM.png

Create an HttpSession:

../Screen%20Shot%202017-02-05%20at%207.06.45%20PM.png

And add an attribute to the session. This works like a map, where the key is a descriptor (“name”) and the value is the worker’s name that the servlet received from the request.

../Screen%20Shot%202017-02-05%20at%207.06.59%20PM.png

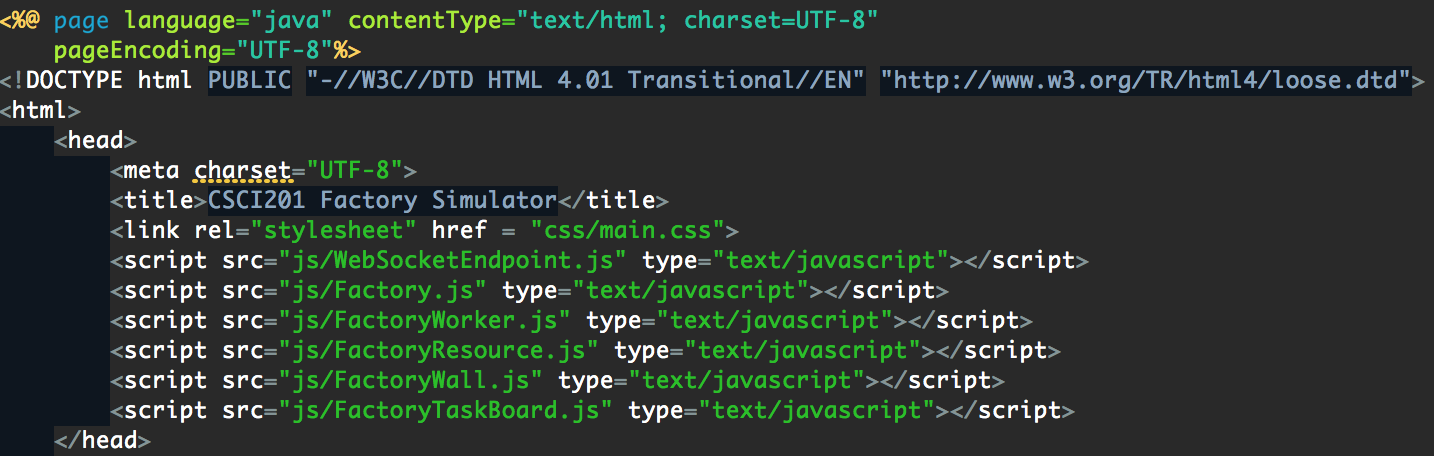
Do the same with the color parameter, and then redirect the site to factory.jsp, which we will write.

../Screen%20Shot%202017-02-05%20at%207.07.14%20PM.png

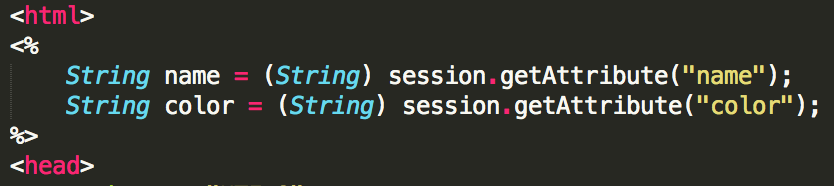
**Part 3 – Dynamic Front End**

We want to use this information in our front end, and also add some custom colors that will require Java, so add a JSP – factory.jsp.

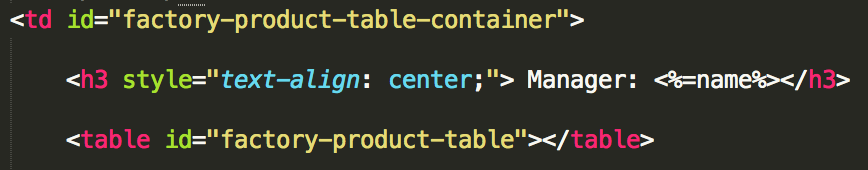
Copy and paste everything in between the <html> tags from index.html



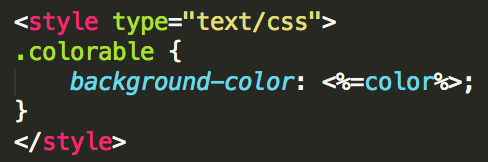
Get the attributes from the session.



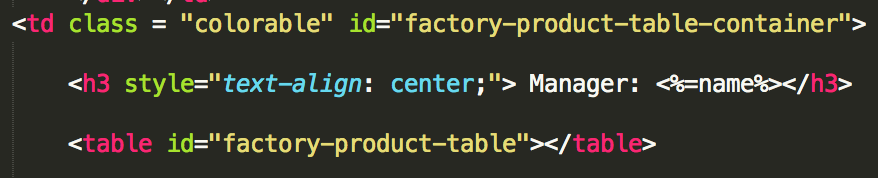
Add a heading tag to display the manager’s name in the factory-product-table-container.

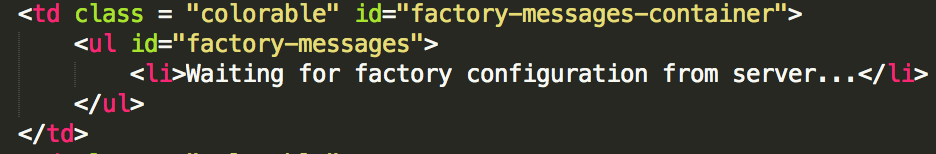


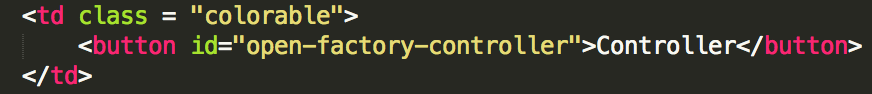
Create a class “colorable” that we will assign to divs that we want to have the selected user interface color.

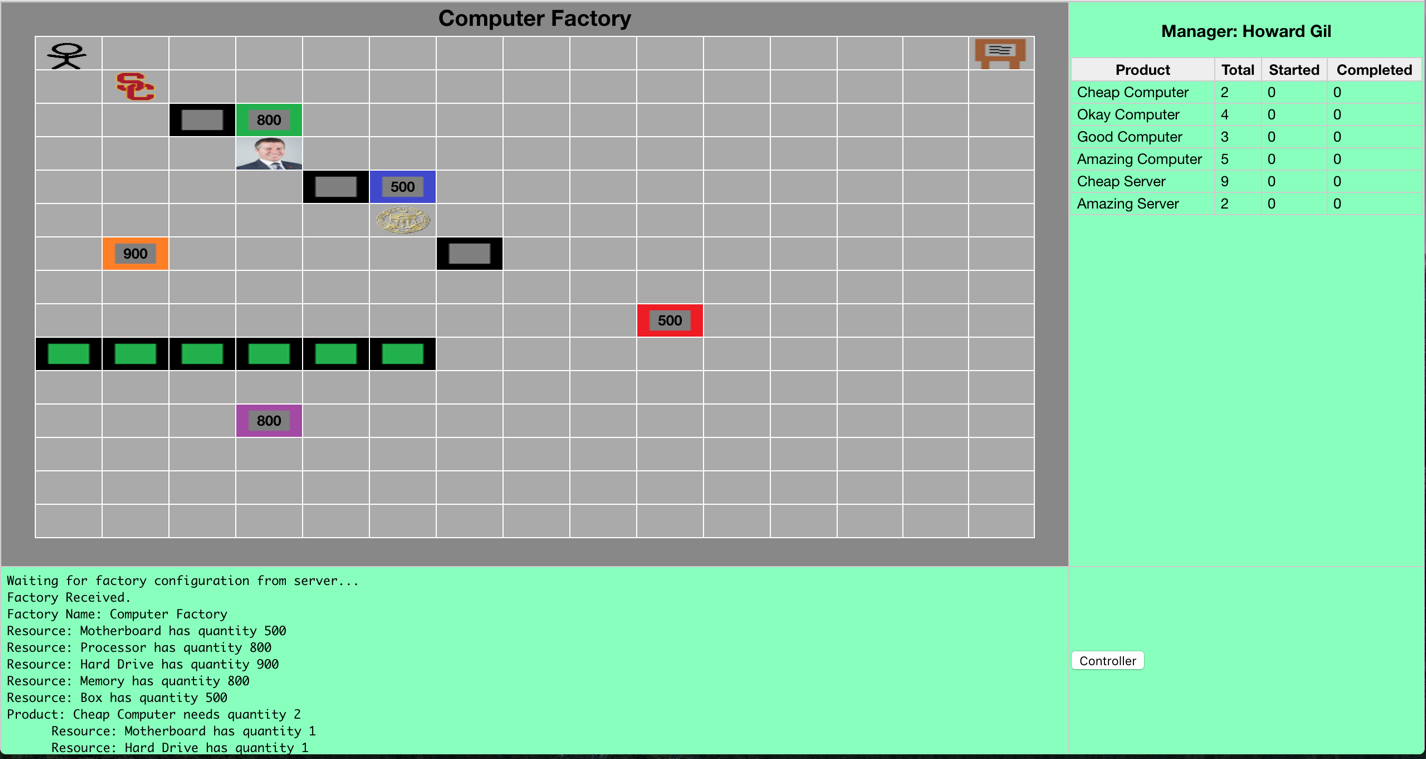


We want to replace our white space with color. Assign this class to the appropriate areas.







If you run setupForm.html and submit a form, you will see this customization.

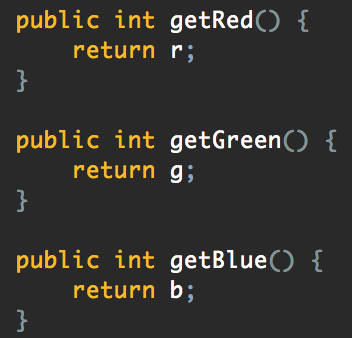
**Part 4 – Using Java in a JSP**

We need more colors. Create a class ColorTool.java in the utilities package. This will generate three random numbers from 0 to 255 to be assigned to RGB ints.

Our random color logic:



Getters for accessing our RGB values:



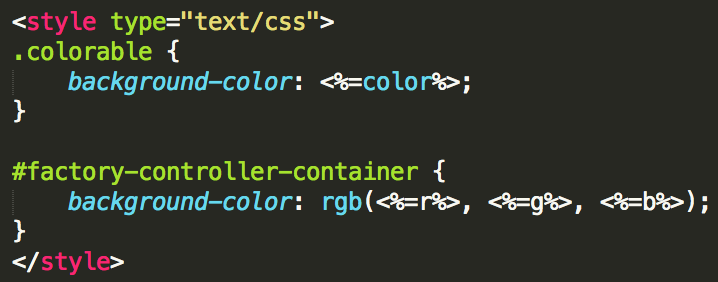
We have to make an instance of this class in our factory.jsp, so let’s import it:

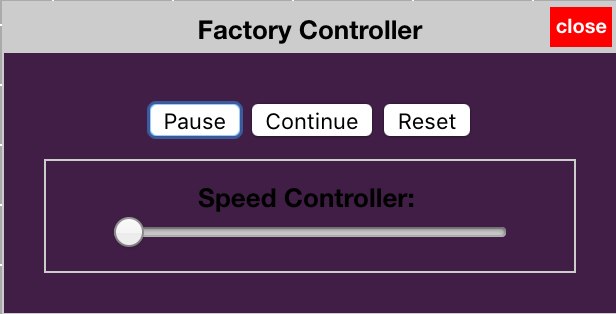
../Screen%20Shot%202017-02-05%20at%205.52.05%20PM.png

And add it to our Java logic:



Now assign this color as the background-color of our controller popup:

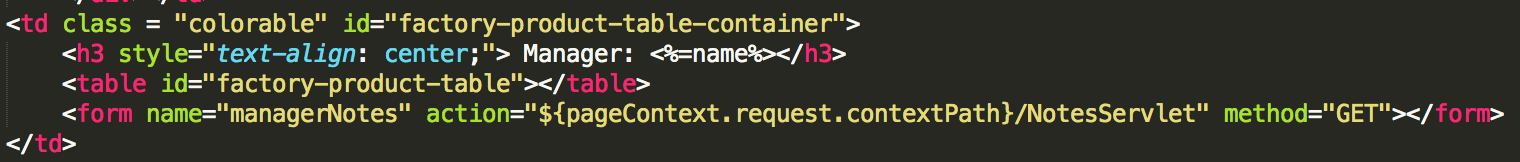


Now our controller window will have a random background color on every run of the factory.

**Part 5 – Using JavaScript for dynamic element creation**

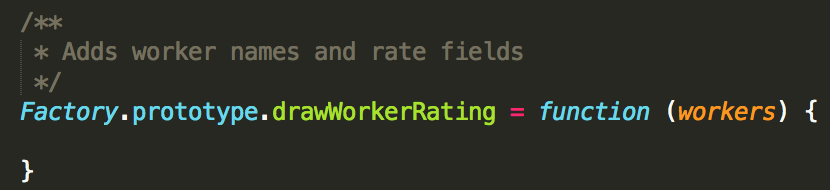
Our factory is not set from the start. It reads and parses factory.txt to determine the number of workers, products, resources, etc. This means our simulation must dynamically add all of these elements at runtime via JavaScript. Let’s add our own dynamic snippet: a form to rate workers’ performance.

Add a form in factory.jsp to hold this. Name it managerNotes.



The actions forwards us to a servlet NotesServlet which you will write in the Expand on This.

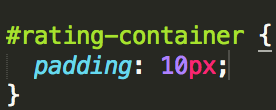
In Factory.js, add a function drawWorkerRating. It takes in a workers parameter, because we want to get the workers’ names from factoryData.



We must create a div container to hold our form elements.

../Screen%20Shot%202017-02-05%20at%206.26.38%20PM.png

Add a rating-container id in main.css just to give our div some padding.



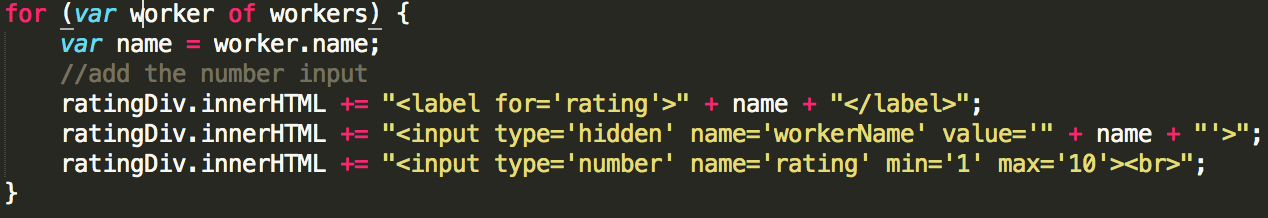
Assign this id to our div.

../Screen%20Shot%202017-02-05%20at%206.26.47%20PM.png

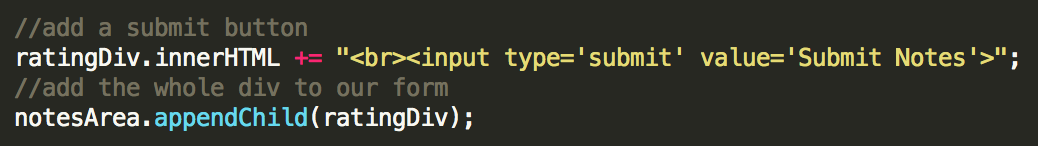
Get the form from our document.

../Screen%20Shot%202017-02-05%20at%206.28.39%20PM.png

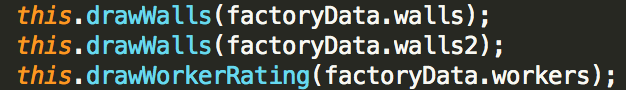
For each worker, we want to add their name followed by a number input field. The name is displayed in a label, but we also add an input of type “hidden” so that the name gets sent as part of the form that NotesServlet receives (we will need this information in the Expand on This). The rating will be limited to values from 1 to 10 so we use the “min”/”max” attributes.



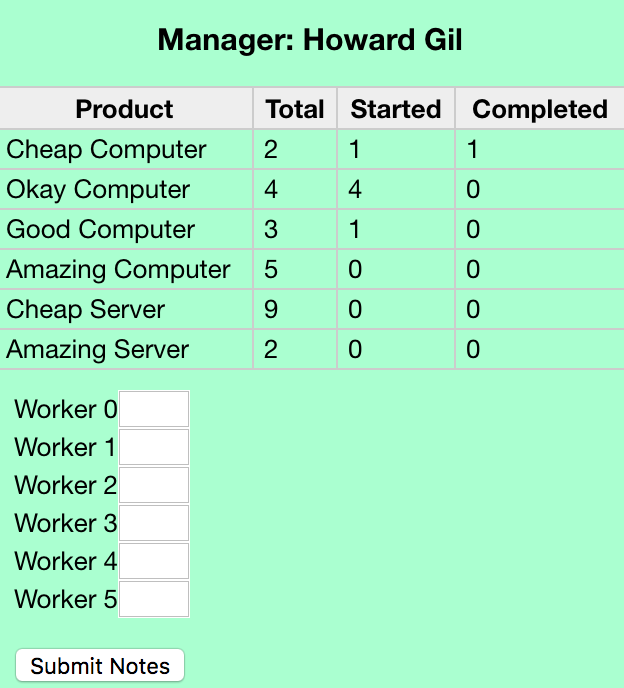
Now add a submit button and add our div to the form.



Call the function in the Factory function.

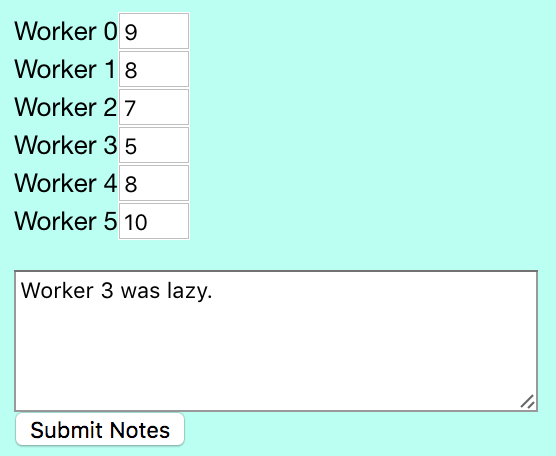


You should now see your form in the factory.

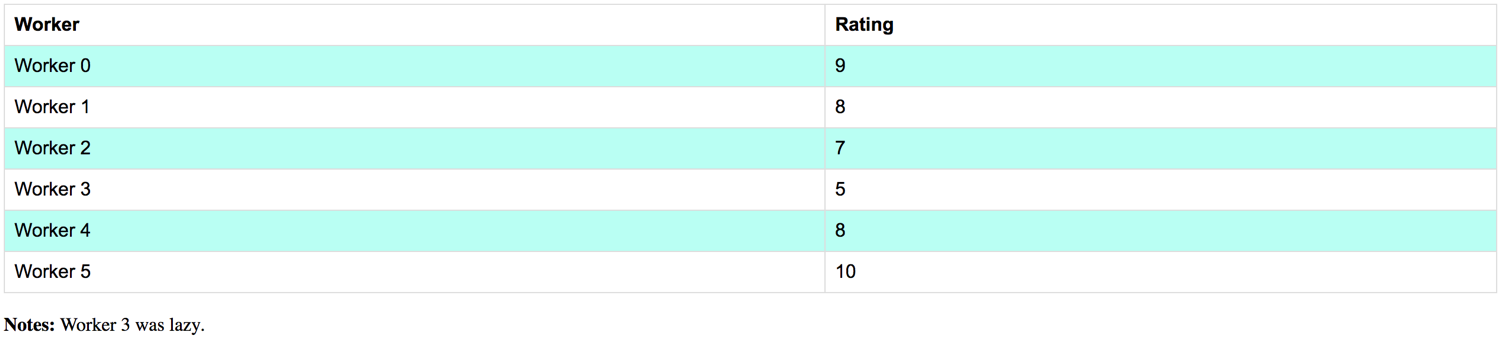


**Expand on This**

Add a text area to the form.



Take this data and pass it to a new JSP, managerNotes.jsp via servlet. Use this JSP to create a table and notes section just like the one below. It must have the same color that was chosen in setupForm.html.



Do not store the names, ratings, or notes in the HttpSession, as they are not variables that are needed throughout the entire “site” (simulation), but only forwarded to the next page.

Instead, set the attributes of the request,

request.setAttribute(...);

And use RequestDispatcher to forward to the managerNotes.jsp.

RequestDispatcher rd = request.getRequestDispatcher("managerNotes.jsp");

rd.forward(request, response);