Report on Web Programming (F28WP) – Coursework 1.

By Daniyar Nazarbayev, H00204990.

## Goal.

The task that was given to us, was to make an e-commerce website, using either servlet or jsp. No external libraries are allowed. The html must follow a xhtml standard. Also, javascript is allowed to be used. The website should have items. Those items should have buy buttons, that put it into a cart.

## HTML/CSS.

Personally, I used the html/css files from my last year's Web Design coursework – an e-commerce website that sells video games. To be exact, I had only made 3 html files – the main page, that has various genres structured as blocks in 3x4 format; clicking any of them will lead to the 2nd page which shows all the games that are part of that genre; the 3rd page is the specific page for the game. That said, only one block in the first page is clickable (MOBAs block), which leads to the 2nd page that has 4 items, again where only one of them is clickable (dota block). The concept is the same, so I decided not to make more pages. That said, for the purpose of this coursework, I did add a few more pages of games to the site map.

I would also like to note that I started the site from ground up, without bootstrap or any libraries whatsoever. I made it on my laptop, which has a 1920x1080 resolution, and it may not look as good on other resolutions. I tried to get the hang of media queries, but unfortunately, writing css code is very tiresome. It requires a lot of checking in between writing your code – trial and error.

## Servlet, JSP.

On to the coursework. I had successfully converted my html to xhtml, and it passes validation without any errors. As for the server side programming, I used the combination of both servlet and jsp. At first I wanted to use jsp only – I found it very weird how in servlet to print html code one had to put it in out(). That said the scriplets turned out to be no better. JSP looked like java, but felt limiting. First of all, they make your code look nearly unreadable.

Second of all, beans – things that are used to transmit data between different pages, are suffering with the same problem. To use a bean, I have to create a class, create variables for it, and then getters and setters. I create a bean and give it a name, use set property to set values, and get property to get them. I guess my laziness took over, because obviously I would much prefer to set all the values with a constructor rather than each value individually.

I researched a lot, and that is how I found out about MVC (Model View Controller) model. Basically, it is when all the computation is done in the servlet, while all the html and scripting is done in jsp. I want to quote this guy on stackoverflow, he said something along the lines of "html code has no place in servlet, and java code has no place in html".

That is what I did, but it was not so easy. The idea was to go from a jsp page to a servlet that would redirect to another cart.jsp page after computation. The problem was that form input parameters can only take strings – any type other than string instantly has a toString() applied to it, and I wanted to pass an object of type Item, that has 10+ variables. At first I wanted to pass all the variables as hidden values, and then use methods such as this to convert them back - Integer.parseInt, Float.parseFloat etc, but then I had other types of variables such as the Gregorian calendar, and string arrays. Then I found out about serialization – the process of converting an object into a byte array. At first I naively converted it

and used a toString(). Turns out the default UTF-8 encoding will lose some bytes when turning it to string, which will affect the object. I had to look for an encoding that does not lose any bytes – one of them was Base64, and it so happened that since Java 8, base64 could be found in java.utils. While Base64 should have worked, it did not, and the reason for that is that the encoding used characters like +, =, / which are also used in URLs. Thankfully for me, it also had an encoder specifically for URLs. After a day worth of research, I was able to send an object as a parameter.

In my Cart.java, which is a servlet, it checks whether the hidden input is null or not, and then decodes the object, and place it into an array of items. In my init() method, I initialized 100 spaces for items[]. I have a method that looks for a null value in there and adds it. The items array is then used when adding to a session.

After that process is complete, cart.java redirects to cart.jsp. There, it checks whether that specific session variable is equals to null, then in a loop, it starts creating a div block for every item. There is a red X button in the corner of every block, and it sends an action to the cart.java. It does the same thing – the button is in a form, which has a hidden encoded object; object is passed, decoded, and then there is a method that looks for that object in the items array. It was a bit tricky, since neither == or .equals would work when checking. After some research I found out that I would have to implement my own methods if I want to check my objects, and those would have to use hash functions (which provide a unique id), or just use a plain id variable (which is what I did). After it finds the first match, it sets that index to null and breaks out of the loop, and then redirects back to cart.jsp.

There used to be one problem with the implementation though. If I were to add 2 different items multiple times in this order:

> “1 2 1”
>
> and would try to remove the last element (1) by clicking the X, it will turn into this
>
> “2 1” rather than this “1 2”

unfortunately, these blocks are printed exactly how they are positioned in the array, and the logic is to delete the first item that matches. To fix that, at first I created a static element that would increment itself, and would be assigned to the id during each constructor call. Unfortunately, everything in the <%! %> block is done only once, just like the init() function in servlet, so I created an extra method that I would call before serialization, which would increment the static variable again and then assign it. The type for that static variable that I used was AtomicInteger. Unfortunately, the normal integer wouldn’t work. From what I read, it has something to do with treading. It says that servlet is “intrinsically multithreaded”, and that static int is incapable of handling multithreading.

That said, this whole implementation has a big drawback – in java, int is only signed and has a max of value at around 2 billion. Every time a page loads, item class gets called, then the method that increments the number and sets it as the id, and after that serialization. If I would refresh the page, it would send another request and the number will get incremented again. I am pretty sure it can easily reach 2 billion, and in a real environment, one would have to restart the server every now and then, to prevent an overflow.

The last part – creating a file with all the purchases works too. The file is created in the eclipse root folder. All the fields have a required attribute, a pattern, and on submit it calls a function confirm().

I wasn't able to validate any of my files – didn't have any time. I did that at the start of the coursework when my html were in xhtml format, but since then I added a lot more to the project. Tomcat 9 was used to run the website, and to get into the main page, you have to go to localhost:8080/Website_v1/html/hub/index.jsp. I wanted to edit my web.xml file, but had no time.

Also, don't forget to check the credits page. As of now, it's only possible to get to credits page in the index folder. I created 3 folders inside my html folder: hub, sub-hub and games. For each of the page in those folders I have to specifically customize the href.

There are about 5 items in the website as of now. I wanted to add more, and there are assets for them, but I didn't have time.