After the exhausting work we put into Project One, it was great to relax for a bit before we started with work on Project Two, which we knew would be just as challenging. To be really honest, Project Two wasn’t that challenging, other than finding the data to input as an XML file. My previous experience with HTML and CSS did help a lot with this project, since it was a database project that employed XML, XSLT and CSS.

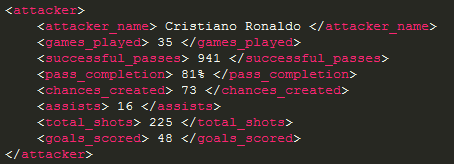
I shall start with how I framed my XML, rationale for my tagging and the hierarchy and the need for my XML amongst users. I will also talk about the extra credit work that I did, but since I borrowed some of that code and adapted it for my own needs, I might be unable to describe exactly how it works.

XML Page

My XML page consists of the statistics of some of the most famous attacking players in soccer today, for their clubs in the 2014-15 season. That is the latest season where the statistics for the whole year are available.

**These statistics are useful for any soccer fan who likes to analyze numbers and compare players that way.**

My tagging rationale is that avid soccer fans search for important contributions to the game by attacking players and my tags are an indication of that.



These indicate the contributions by that certain attacker, including the name. So, this makes the search easier, and more relevant.

The hierarchy is in a certain format, and the first tag is the name of the player himself, followed by the number of games played, since that is one of the most important statistics.

The rest of the statistics, including the goals scored, are a realization of the player as a true attacking talent.

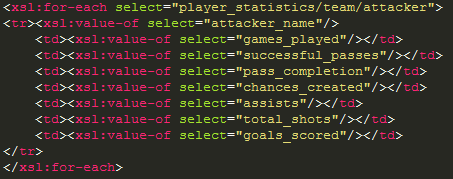
The hierarchy is as follows –

The most important tag is ‘player\_statistics’ because that determines what the XML file is all about. The next important tag is the ‘team’ tag, since the players belong to different teams. The next tag is the ‘attacker’ himself, since all the child nodes within this tag are his own statistics. This hierarchy was one which seemed most sensible to me.

I wanted to create separate tables for the players from each team, but that would mean that the XSL code would require a lot of code, and I was unsure of how to do it. So, I ended up creating this, and it is quite helpful for anyone who wants statistics for the top attackers in world soccer.

XSL Page

My XSL page makes basic use of an xsl:for-each loop and multiple quotes of xsl:value-of within that loop to pull from the XML page and display as an HTML page.



All these are input in the form of a table, and each of the XML values displays under table headers.

The link to the CSS file is also coded into the ‘head’ part of the XSL code.

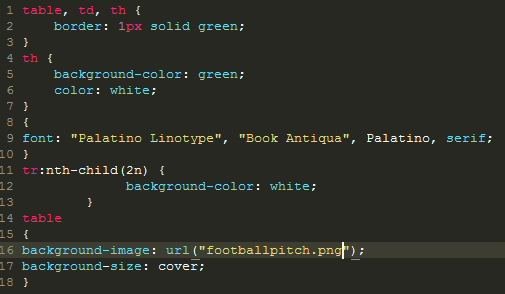
CSS file

The CSS file is my most basic file of the lot and the least well thought out, since my implementation of styles isn’t the best.

I’ve included 5 styles or slightly more, since I wanted to work on my extra credit, and this is probably why my CSS file has suffered.

I found the CSS file the easiest one to work on, because there’s no Declaration line, any kind of framework other than the definition of the Styles themselves.

The only clever part of my CSS file is the Nth Child Exception for coloring alternate lines, and the forced fitting of a background in my table.



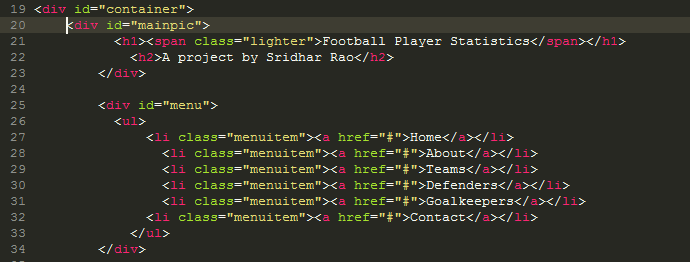
This is the extent of my CSS file.

Extra Credit

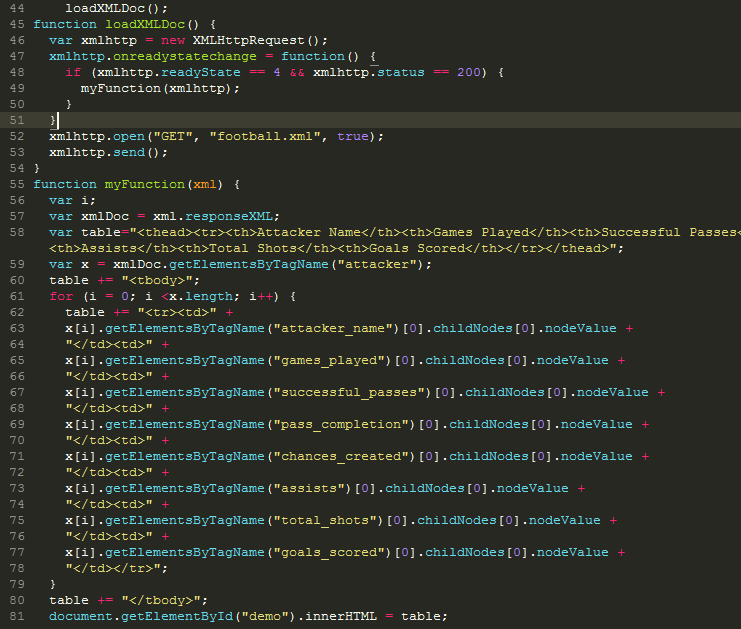
I managed to do two parts of the extra credit, but it is code that I reused and adapted to my own needs.

I used Javascript to pull XML data from my page into a table, essentially creating an HTML page. I also created a header with a navigation bar and a footer for my HTML page by using old code from a website.

This is a sample of the Header + Nav Bar –



This is a screenshot of the Javascript code that pulls from the XML file to create a table –



This code was probably some of the most complicated of this project, but because I have done C++ and Java before, I could make some sense of this and use it to my advantage. The footer was quite simple compared to the Javascript part, but I still did it.

I hope including the two extra credit sections inside one file is okay.

I enjoyed this project because it had very little to do with reading and a lot to do with experiencing coding, which isn’t my most favorite thing in the world, but it is still interesting.

I loved the XML portion of the project, because it gives us a sneak peek at Single Sourcing on a tiny scale, but also shows us how to implement it. This has prepared us for life after a degree.