PA3 Writeup

In order to create my web crawler, the first thing I did was create a method that allows the user to input the URL of the website they want to crawl in the format of [www.cnn.com](http://www.cnn.com). Once the root url of [www.cnn.com](http://www.cnn.com) has entered, I put the robot.txt of the website ([www.cnn.com/robots.txt](http://www.cnn.com/robots.txt)) into my queue. In addition, I hard-coded BleacherReport’s robots.txt to enter the queue along with cnn.com. In addition, it sends a start message through another queue. Once a robots.txt and start message has been received, my worker role starts to read the sitemap through recursion. During this “loading” stage, the visited url and disallowed urls are both kept tracked in one of my classes called visitedUrlCheck. In order to parse the sitemap, I use LINQ to remove everything but the http links. If the http links contains “.xml”, then we continue the recursively parsing of the sitemap. Else, the http link is put into a queue, which will be accessed during the “crawler” stage. The only restrictions placed upon the sitemaps are that it hasn’t been visited before and that it is newer than April 1st, 2015.

Once the “loading” stage has finished, then the “crawling” stage begins. For each http link, I first use a check to make sure it is not disallowed or already visited. If that check passes, then I use LINQ to find the title, other links and last modification date if it exists. Once I have this information, I put the link inside the table storage along with other key information like index size, total amount of URL crawled, CPU usage and Memory Available. On the other hand, I created a new table that helps me catch errors if they exist.

For my dashboard, I use AJAX to communicate my HTML with my WebService1.asmx.