**URL Change Detector:** The goal of this console application is to take a list of urls and download contents from them testing for changes. We will internally be using this tool to monitor as part of an automated test suite (TeamCity) that will run the console application frequently test the contents of thousands of urls so the application needs to be reasonably fast and robust in its error handling.

* **Console application input parameters**
  + **Parent Directory - String (optional)** - the directory to run in (default to using the directory the console application is located in)
  + **Threads - Int32 - (optional)** - the number of threads to use while running (default to 1)
  + **Replace Domain - String (optional)** - defaults to empty string, if a non null / empty value is supplied then for each url in the list the domain name (ie abc.com) will be replaced with the base url value. (this allows a list of urls to be pointed at different test servers with low effort)
  + **Keep Old Runs - Int32 (optional)** default to 2 - the number of Current Run folders to retain (keep the most recent ones). Older current runs should be checked for and deleted on application execution.
* **Input File File Columns** (in Tab Separated value format that is expected to be in the Parent directory and named InputURLS.txt)
  + URL: the url to contact
  + METHOD: GET or POST
  + POSTCONTENT: if post is selected this is the contents to post

Example content (first row is header row

URL METHOD POSTCONTENT

<http://abc.com/test> POST {“hello”:”world”}

* **On execution the application should**
  + **Read a list of urls from InputURLS.txt in the parent directory**
  + **Create a Folder for the Current Run in the Parent directory whose name is a human readable date time like 2016\_12\_02-0933-01**
  + **Create a log file URL\_Results.txt in tsv text format in the current run folder that we will append result info to for each url**
  + Check to see if a one or more previous current run folder(s) exist, if so then grab the name of most recent one and we will use it for comparison below, and delete any older versions found beyond the KeepOldRuns value
  + **For each url (this would run as a parallel.foreach with a max degree of parallelism set to the threads value from the input parameters)**
    - **Download each url and save its results to a file whose name is a numeric hash of the url (ie string.gethashcode or something similar)**
    - **Log the total time to download the file**
    - **Log the success or failure (including any error messages)**
    - **If that url was present in the previous run, compare the results to the prior run, and log it the two were identical or different**
      * **If different scan through both files line by line using a StreamReader (we will blindly be treating all files as text regardless of the response type) and count the number of identical lines, and stop on the first different line, and log the line number and the first 100 characters that differ (log both the expected and the actual)**
  + After processing all urls the system should report back the total run time, the total success and fail count to the console and log those metrics to Current Run Folder in a file named OverAllStats.txt