

Sayeed Ahmed – SEC01 (NUID 002191535)

# Big Data System Engineering with Scala

## Spring 2023

### Assignment No. 6



Implementation:

WebCrawler.scala:

```
def wget(url: URL)(implicit ec: ExecutionContext): Future[Seq[URL]] = {
  // Hint: write as a for-comprehension, using the method createURL(Option[URL], String) to get the
  // 16 points.
  def getURLs(ns: Node): Seq[Try[URL]] = {
    for (href <- (ns \\ "a").map(_ \\ "@href")) yield validateURL(createURL(Option(url), href))
  }

  // Hint: write as a for-comprehension, using getURLContent (above) and getLinks above.
  // 9 points.
  for {
    content <- getURLContent(url);
    urls <- MonadOps.asFuture(getLinks(content))
  } yield urls
}
```

MonadOps.scala

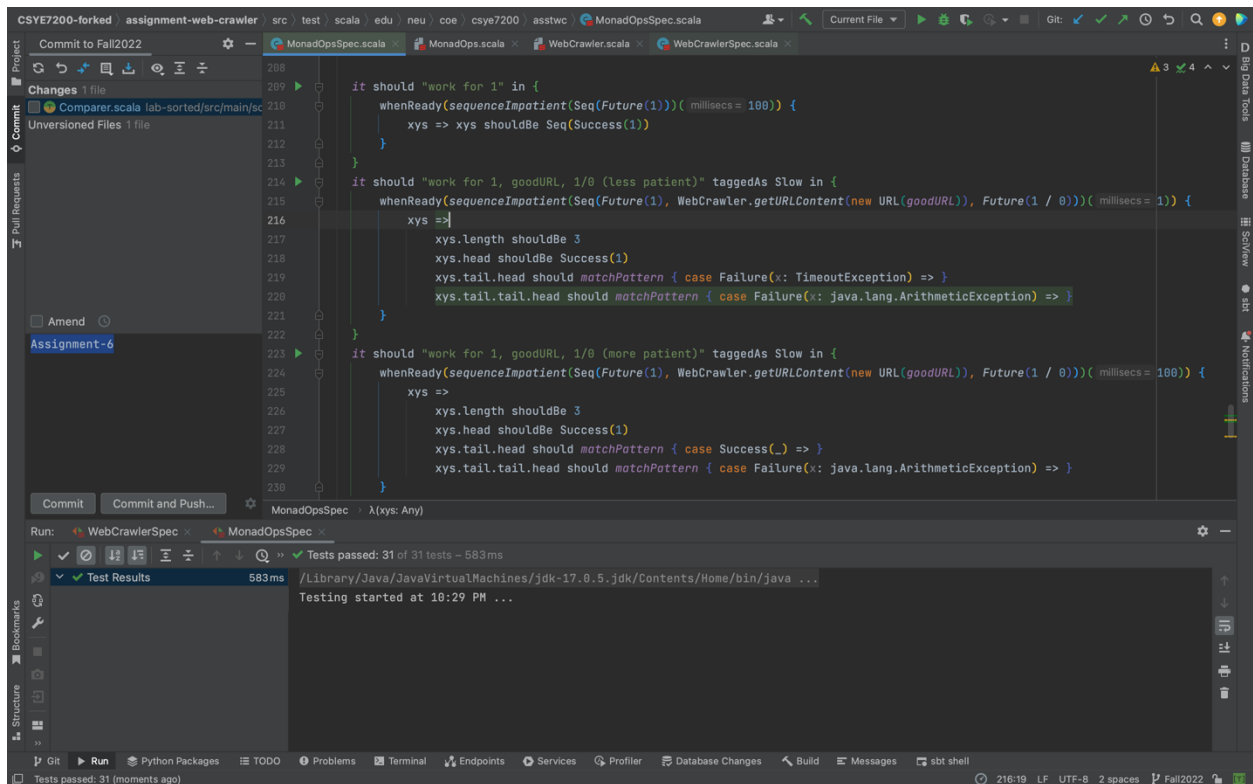
```
def asOption[X](xe: Either[Throwable, X]): Option[X] = xe.toOption // TO BE IMPLEMENTED
```

github link: <https://github.com/sayeedahmed01/CSYE7200/tree/Fall2022/assignment-web-crawler/src/main/scala/edu/neu/coe/csye7200/asstwc>

Suggestions:

- 1) Implementing a delay: Currently doesn't have a delay, implementing a delay between requests can prevent overloading servers and being identified as a spam bot.
- 2) Handle errors: Current implementation outputs errors to the console and doesn't take any action. A better approach would be to retry and mark them failed after a few retries.
- 3) Use a cache to avoid redundant requests: We can implement a cache to store the HTML content to avoid redundant requests.

Test:

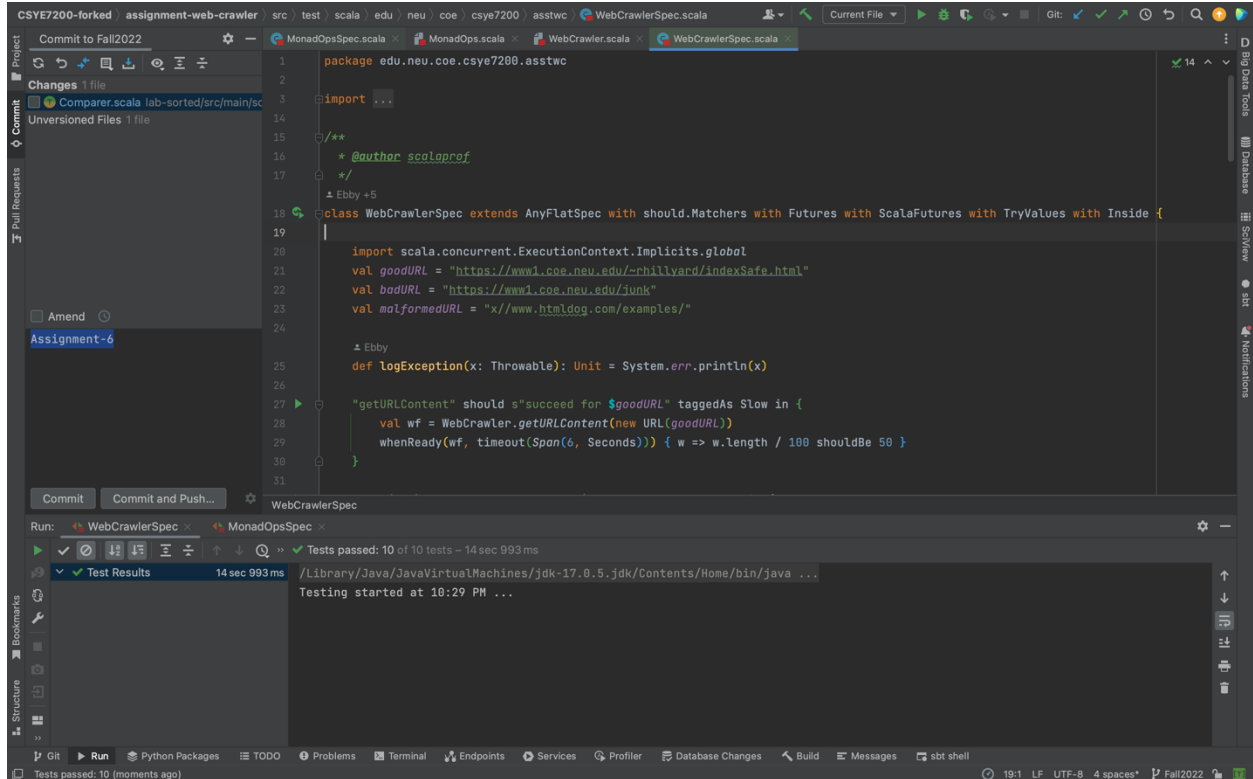


```
Commit to Fail2022
Changes 1 file
  Comparer.scala lab-sorted/src/main/sc
Unversioned Files 1 file
Pull Requests
Amend
Assignment-6
Commit
Commit and Push...
Run: WebCrawlerSpec x MonadOpsSpec x
Tests passed: 31 of 31 tests - 583ms
Test Results 583ms
Testing started at 10:29 PM ...
Testing started at 10:29 PM ...

it should "work for 1" in {
  whenReady(sequenceImpatient(Seq(Future(1)))( milliseconds = 100)) {
    xys => xys shouldBe Seq(Success(1))
  }
}

it should "work for 1, goodURL, 1/0 (less patient)" taggedAs Slow in {
  whenReady(sequenceImpatient(Seq(Future(1), WebCrawler.getURLContent(new URL(goodURL)), Future(1 / 0)))( milliseconds = 1)) {
    xys =>
      xys.length shouldBe 3
      xys.head shouldBe Success(1)
      xys.tail.head should matchPattern { case Failure(x: TimeoutException) => }
      xys.tail.tail.head should matchPattern { case Failure(x: java.lang.ArithmeticException) => }
  }
}

it should "work for 1, goodURL, 1/0 (more patient)" taggedAs Slow in {
  whenReady(sequenceImpatient(Seq(Future(1), WebCrawler.getURLContent(new URL(goodURL)), Future(1 / 0)))( milliseconds = 100)) {
    xys =>
      xys.length shouldBe 3
      xys.head shouldBe Success(1)
      xys.tail.head should matchPattern { case Success(_) => }
      xys.tail.tail.head should matchPattern { case Failure(x: java.lang.ArithmeticException) => }
  }
}
```



```
Commit to Fail2022
Changes 1 file
  Comparer.scala lab-sorted/src/main/sc
Unversioned Files 1 file
Pull Requests
Amend
Assignment-6
Commit
Commit and Push...
Run: WebCrawlerSpec x MonadOpsSpec x
Tests passed: 10 of 10 tests - 14 sec 993ms
Test Results 14 sec 993ms
Testing started at 10:29 PM ...
Testing started at 10:29 PM ...

package edu.neu.coe.csye7200.asstwc

import ...

/**
 * @author scalaprof
 */
class WebCrawlerSpec extends AnyFlatSpec with should.Matchers with Futures with ScalaFutures with TryValues with Inside {

  import scala.concurrent.ExecutionContext.Implicits.global
  val goodURL = "https://www1.coe.neu.edu/~chillyard/indexSafe.html"
  val badURL = "https://www1.coe.neu.edu/junk"
  val malformedURL = "x/wwww.htmlldog.com/examples/"

  def logException(x: Throwable): Unit = System.err.println(x)

  "getURLContent" should s"succeed for $goodURL" taggedAs Slow in {
    val wf = WebCrawler.getURLContent(new URL(goodURL))
    whenReady(wf, timeout(Span(6, Seconds))) { w => w.length / 100 shouldBe 50 }
  }
}
```