Scala Concurrency

Practical Exercises

Chapter 1: Basic Concurrency with Scala

1. A PowerPoint source file, ending with the suffix .pptx, is actually a zip archive file containing a hierarchy of XML files, each of which describes some aspect of the presentation. You will find the PowerPoint source files for this course, and you can examine the contents of these files manually using an unzip utility.

   The Java classes in the package `java.util.zip` allow these files to be processed, and these classes can of course be used from Scala programs. In particular, the class `java.util.zip.ZipFile` represents a zip archive file, and provides a number of methods that allow it to be read and analysed. You can find out more details of this in the appropriate JavaDoc API page.

   The method entries on a `ZipFile` instance returns an (Java) enumeration of the entries contained in the zip archive. We can access this as a Scala iterator simply by importing (implicit) conversions using the following:

   ```
   import scala.collection.JavaConversions._
   ```

   Each slide in the presentation is represented by a file in the (archived) directory `ppt/slides`, with name **`slide`*`nn`*`.xml`**

   Write a function that returns the number of slides in a given presentation, by counting these files. Write a further function that calls this for each of the `pptx` files in the attached directory, and returns the total number of slides in all the files.

   Now refactor the counting function so that instead of returning an `Int` to represent the number of slides, it returns a `Future[Int]`, and runs asynchronously. Invoke these functions so that we can still return the total number of slides.

   If possible, measure the time taken to perform both the initial, synchronous, version and the modified asynchronous version and (hopefully) illustrate a difference!