Intermediate Scala

Practical Exercises

Chapter 1: Functional Exception Handling

1. The class **`java.net.URL`** is used to represent a resource that is identified by a URL. It may be a local resource such as a file, or a remote resource accessed using http or some other protocol. Write a Scala function/method, **`URLFromString`**, that attempts to convert a String into its corresponding URL object. Note that if the constructor for the URL class cannot recognise the String as a URL, it will throw the **`java.net.MalformedURLException`**. So the **`URLFromString`** method should return a **`Try[URL]`** value that encapsulates both cases.

   Ensure that your method behaves as expected.

2. Once we have a URL object, we can attempt to connect to it and retrieve its contents. The methods **`URL.openConnection`** and **`Connection.getInputStream`** are used to achieve this.

   Extend your function from above so that, given a String representing a URL, we can return an object of type **`Try[java.io.InputStream]`**, which will contain either an instance of the **`InputStream`** that we can read, or an exception that details why this was not possible.

   Your function will include multiple nestings of **`Try`** types, so you may wish to investigate **`flatMap`** as a way of dealing with this.

   Note that in order to connect to an http based URL you may need to configure a Proxy object, or supply authentication information. Alternatively you can use a local file, specifying its URL as

   `file:///path/to/file`

3. Finally, define a function that builds on the above, returning a **`Try[Iterator[String]]`**, where the Iterator takes us through each line of content in the URL if successful, with the Failure branch of the **`Try`** containing the exception that caused the operation to fail.