SWEN30006 Project 1

Getting To Grips With Ruby

The Task

Your task is to develop a **Ruby** program that will download news articles from a variety of JSON and RSS (XML) sources and store these articles in an independent data format. You will then be required to export the downloaded data in JSON, CSV and XML formats.

This project will act as the first stepping stone towards the implementation of a news notification system in Project 3, provding you with the tools you will need to download and store your news articles.

Program Structure

To assist in your implementation of the project requirements, we have provided both a sample dummy project implementation and a module `News` that you will need to use in your program. This will likely be the first time you have used a module in ruby, so we will start by describing how you use this module and the function of each class.

A module defines a namespace in Ruby. A module can include functions, variables, classes and further modules. In this instance, we are using the module to namespace our provided news classes in one wrapper. To access these classes, you prefix the class with the namespace but otherwise use it as normal. For example, to instatiate an Article, you simple call:

```
article = News::Article.new(arguments)
```

Important Note You will **not** need to modify any of the files within the news module for your submission, and we will be using our own version of the news module when testing your program for submission. You will however be expected to extend these classes in your implementation.

When extending classes, you should make sure that you define the initialise method and also call super within that initialise method so that the checks on subclasses are run.

News `news.rb` The news file is the file that describes the base module, and imports all the relevant files into the program. To include it, you simply need to `require_relative 'news.rb'` from your program. This will then import all of the other files for use in your program.

News::Article `news/article.rb` The news article class contains the basic data structure for a news article with instance variables to store the information from a basic xml news feed. This class uses named method paremeters to pass in arguments to initialise. You can initialise the articles like so:

```
article_one = News::Article.new(author: 'Mat', title: 'Dummy',
    summary: 'Some really long string',
    images: 'www.some_image.com', source: 'Canned Data',
    date: Date.today)
```

This provides semantic meaning to method arguments and makes it easier to read. In your project you will be expected to to create subclasses of this class to store extra information provided by your chosen sources.

News::Exporter `news/exporter.rb` This class takes a News::Importer class and News::Formatter object and will export a given array of articles in the format specified by the News::Formatter object. You **should not** modify this class for your project, but you may wish to understand what is happening in the file for your own interest and understanding of how the program interacts.

News::Formatter `news/formatter.rb` This class forms the basis for an output format for your program. This is one of the two main classes you are expected to extend and implement methods of in your project. The class does not do anything itself, but instead forms an interface for your subclasses to implement. The methods you will be required to implement are the following:

- `extension`
  - This method should return the extension for the output file (i.e. 'json','csv' or 'xml') to be appended to the output file name. This must be a string.
- `header?`
  - This method should return true or false as to whether this output format requires a header to be printed before the body of the news artciles.
- `header(article)`
  - If the format returns true to `header?` this method should return a string (including line breaks) that define the header of the output document. The method should take an article as an input argument (though does not need to use it).
- `footer?`
  - This method should return true or false as to whether this output format requires a footer to be printed before the body of the news artciles.
- `footer(article)`
  - If the format returns true to `footer?` this method should return a string (including line breaks) that define the footer of the output document. The method should take an article as an input argument (though does not need to use it).
- `article_representation(article)`
  - This method takes an article as an input argument and returns a string representation of the article in the prescribed format. This should include any required indentation before the string if needed.

As part of this project, you will be required to extend this class for each export format for each source you have.

News::Importer `news/importer.rb` This class forms the basis for an importer for your program, you are expected to extend and implement the methods required for each of the three sources that you have chosen.

This class defines an initialisation method that takes a `start_date` and `end_date` as input arguments. These variables define the limits of articles you should include from the scrape.

The news importer also includes an array of articles `@articles` and a method `articles` which return unique articles from the `@artciles` array. In your extension you must implement two methods:

- `self.source`
  - A class method that returns the source name as a string. This must be defined as a class method.
- `scrape`
  - An instance method that is responsible for performing the actual scrape and storing articles in the `@articles` array. This method will be called only once, and should only return once all of the articles have been added to `@articles`.

Note that while your scraping will be run through the calling of one method, you should **not** restrict yourself to putting all of the scraping logic in this one method. We want to see good functional decomposition and good separation of concerns (i.e. scraping and interpretting) in your submissions. Failing to do so will loose you marks for poor code quality.

News::Scraper `news/scraper.rb` This class takes several importer/formatter pairs, along with a source name, and is then responsible for running each importer and outputing the data with the registered export format. It will provide a start_date of 7 days ago, and an end date of today, for the import (thereby scraping the past week's worth of data).

You will not need to modify this class for your project, but you will need to use it in your driver script. The provided example `dummy_project.rb` demonstrates appropriate use of this class. You may be interested in looking at the source code though to understand what is happening.

On Plagiarism We take plagiarism very seriously in this subject. You are not permitted to submit the work of others under your own name. This is an **individual** project. More information can be found here: (https://academichonesty.unimelb.edu.au/advice.html).

Using a JSON API

JSON (Javascript Object Notation) is a commonly used web data format that allows the transmission of structured data across platforms and languages. Here we will be interacting with news APIs that use JSON to submit requests and return JSON data as a response.

To do this, we will use the ruby Net::HTTP library to make requests, and the JSON library to parse the response body.

```
require 'open-uri'
require 'json'
require 'net/http'
```

```
 Define the URL
URL = 'http://www.matblair.com'

 Define the HTTP object
uri = URI.parse(URL)
http = Net::HTTP.new(uri.host, uri.port)
 If the api being scraped uses https, then set use_ssl to true.
 http.use_ssl = true

 Define the request_url
request_url = '/resume.json'
 Make a GET request to the given url
response = http.send_request('GET', request_url)

 Parse the response body
forecast = JSON.parse(response.body)
```

The Net::HTTP class can also be used to make post and put requests. The documentation of this class can be found here: http://docs.ruby-lang.org/en/2.0.0/Net/HTTP.html. You should refer to the specific API documentation for your chosen source.

Parsing an RSS Feed

RSS is a feed format originally specified in 1999 and built on since that allows for easy syndication of news articles between sources. The full specification can be found here: http://www.rssboard.org/rss-specification.

RSS is commondly found from most large news sources or publishing houses, whilst it does not provide as much information as most JSON apis and has limitations in accessing archival data, its prevelence makes it an important tool that cannot be ignored for the application you will be building throughout these three projects.

Ruby has built in language support for parsing RSS feeds that will parse any correctly formatted RSS 1.0 and 2.0 feed and allow you to iterate through each of the articles in the RSS feed and access their attributes directly. The following example highlights how to pull down and parse the results from the news.com.au national feed and print the title, publication date and link to the full article.

```
require 'rss'
require 'open-uri'

url = 'http://feeds.news.com.au/public/rss/2.0/news_national_3354.xml'

open(url) do |rss|
    feed = RSS::Parser.parse(rss)
    puts "Title: #{feed.channel.title}"
    feed.items.each do |item|
        puts item.title
        puts item.pubDate
        puts item.link
    end
end
```

Limitation of RSS Feeds RSS feeds, by their nature, are designed to be checked every 'x' minutes. The feeds include a value, the `ttl` field, which specifies how frequently a source should be refreshed.

For the purposes of this first project we will only be expecting that you export the current contents of the rss feed at any given time. For Project 2 and 3, we will be storing the data as we read it in the database, and therefore will be able to store the information as we read it and gain more usefulness from these RSS feeds.

Storing News Articles

You should create instances of `News::Article` (or the appropriate sub class) as you scrape information from either the RSS or JSON sources. You should not use any other data structure to store this information. This will mirror the structure that you will be using in Project 2 when you transfer this to a Rails project.

Sources

In this project you will be building the base layer for an eventual news digest service. Therefore, we will need a variety of news sources to be able to deliver a comperhensive weekly digest. We will be requiring that you scrape atleast **one JSON API and one RSS source**. Your third choice may be either, it is up to you.

Here we provide a list of sources that you may choose to scrape from. Note that all sources offer roughly the same amount of information, and your marks will not be impacted by which you choose as long as you have atleast one JSON and one RSS source. Therefore, you should pick sources that interest you.

RSS Feeds

Below is a selection of RSS feeds for local news sources. You may pick any of the RSS feeds offered by these news articles. In addition, you may pick the RSS feed of a source you read frequently as your third choice, provided that the content is appropriate.

- The ABC: http://www.abc.net.au/services/rss/
- The Age: http://www.theage.com.au/rssheadlines
- The Herald Sun: http://www.heraldsun.com.au/help/rss
- The Sydney Morning Herald: http://www.smh.com.au/rssheadlines
- SBS: http://www.sbs.com.au/news/rss-list

JSON Sources

For sources that require a search term, you should pick a term that is relevant to your interests. The content of the news articles is not important at this stage.

- The New York Times
  - Instructions: http://developer.nytimes.com/
  - Description: The New York Times offers a wide range of news apis to search articles, though it is primarily focussed on US news so will not have any local content.

- The Guardian
  - Instructions: http://open-platform.theguardian.com/access/
  - Description: Similar to the New York Times, The Guardian offers API access to all of their content from 1999 onward. They provide free access to their developer api for up to 12 calls a second.
- Webhose.io
  - Instructions: https://webhose.io/
  - Description: A web aggregation service that searches hundreds of sources to provide structure keyword responses to API calls and return data. They provide free access for up to 1,000 requests per month.

Submission Instructions

Project submission will be enabled through the LMS. We will require you upload a zip file of your project directory including the skeleton code. You do not need to include the example project. This will then be run against automated tests for correctness. We will be running your program like so:

```
ruby 613625-project-one.rb
```

Your main project-one.rb file should be a ruby script (like `dummy_project.rb`) that instatiates a `News::Scraper` object, adds the importers and formatters to that object, and calls scrape on that object.

You should also ensure that you make proper use of `require_relative` to load any files you need into your program. Failing to do so may result in your program not running depending on where we run your project from.

Further, your project should not rely on any other gems, only the standard ruby library methods may be used. If your project relies on other third party gems it will fail automated marking and you will get 0 for the the importers, formatters and news item subclasses.

Implementation Checklist

- Defined 3 different Importers for three difference sources.
  - Defined atleast one JSON importer
  - Defined atleast one RSS importer
- Defined 3 output formatters for each source
  - Defined a JSON formatter for each source
  - Defined a XML formatter for each source
  - Defined a CSV formatter for each source
- Defined at least one News::Article subclass with additional data from your sources.
- Defined a driver program that adds all the importers and formatters to an instance of News::Scraper and runs the scrape process.
- Tested your program as you will submit it.

Marks

This project will account for 10 marks out of the total 100 available for this subject. These will be broken down as follows:

Criterion	Percentage
3 different importers (atleast one JSON and one RSS)	3%
JSON, XML and CSV output formatters	2%
Code Quality (decomposition, variable naming, use of constants, ruby idioms)	4%
Suitable News Article Subclasses Defined	1%

We also reserve the right to award or deduct marks for clever or poor implementations on a case by case basis outside of the prescribed marking scheme.

The code quality will be judged on how well you have adapted to Ruby style. We expect to see good variable names, well commented functions, inline comments for complicated code. We also expect good object oriented design principles and functional decomposition. Output correctness will be automatically verified against expected results, therefore it is important your program outputs the correct format.

To assist with your code quality, we suggest you review the following ruby style guide: https://github.com/bbatsov/ruby-style-guide. We will be using this style guide to form the basis of our style marking.

If your project fails the automated tests, you will get, at most, 1/2 marks for the importers, output formatters and news items.

Submission Date

This project is due at **11:59 p.m. on the 24th of August.** Any late submissions will incur a 1 mark penalty per day unless you have supporting documents. If you have any issues with submission, please email Mat at mathew.blair@unimelb.edu.au, before the submission date.