CSE/ISE 337: Scripting Languages

Stony Brook University

Extra Credit Assignment

Fall 2020

Assignment Due:
Wednesday, Nov 4th, 2020, by 11:59 PM (EST)

Learning Outcomes

After completing this programming assignment, you should be able to:

- Scrape medium-sized web pages.
- Read and comprehend basic HTML and CSS.
- Use external Python modules.
- Interact with Python's File IO.

Instructions

- Read the problem description carefully.
- Explore the links provided. They are critical to understanding which elements and attributes are relevant.
- Pay attention to where you write the output of your script (e.g., directory names and files names).
- Do not leave unnecessary print statements in your script.
- Submit one file called **android-crawler.py**
- Your submission will be graded as per test cases. You will get credit for every passing test.

Problem Description (50 points)

Background

Android is one of the most widely used mobile platform. One of its hallmarks is that it allows users to extend its functionality by developing apps that can run on it. Android provides numerous Application Programming Interfaces (APIs) to facilitate app development. The set of APIs is officially maintained by the contributors to the Android platform. The documentation of these APIs is maintained on Android's official website called the Android API reference (https://developer.android.com/reference). This website is very useful for app developers since they can refer to it while developing apps. The reference is divided into a collection of packages. Each package offers some classes and interfaces that encapsulate certain features of the Android platform. For example, the **android.app** package (https://developer.android.com/reference/android/app/package-summary) contains classes and interfaces that encapsulate the overall Android application model and the fundamental building blocks of an Android app (e.g., activities and services). These packages are vast. A user may often get lost when trying to lookup a particular aspect of an API. For example, the documentation contains many APIs that have special cautionary notes attached to them. These notes serve as warnings to the user of the API. As an example consider the *newKeyguardLock* method in the *KeyguardManager* class (https://developer.android.com/reference/android/app/KeyguardManager#newKeyguardLock(java.lang.String)). This method has a note of warning or caution associated with it that says that the method was deprecated in a particular version of Android and should not be used without providing a mechanism for backward compatibility.

Assume that a developer wants to find out all methods and fields in a class or an interface in the **android.app** package that has a note of warning. Since, this would entail a huge amount of manual labor, we need to help the developer by automating this task. Since the entire documentation of this package is on the web, we will write a web scraping script to automate this task.

Task to be Performed

Look at the **android.app** package (https://developer.android.com/reference/android/app/package-summary). You will see a list of interfaces, classes, and exceptions. Some of these list entries have methods or fields that have a special *cautionary note* associated with them. Write a script to do the following:

1. For every interface, class, and exception in **android.app** package, identify the method or field names that have a special cautionary note attached to it. For example, the *getRunningTasks* method in the *ActivityManager* class has a cautionary note (https://developer.android.com/reference/android/app/ActivityManager#getRunningTasks(int)) that says this method is not available to third-party apps since it can leak personal information to the caller.
2. For each such entity that you identified in step 1, save them in a file. The name of the file should be the corresponding interface name or class name or the exception name. Every entity that you find in step 1 should be saved in the format **<api-name>:<note-text>**. The file name and format should be strictly adhered to.
3. Save all files in a directory called **outFiles** in the same directory as your script.

As an example, take the *KeyguardManager* class in the **android.app** package. Since this class has 4 entities that have a cautionary note attached, the script should create a file **outFiles/KeyguardManager** with the following lines:

- *createConfirmDeviceCredentialIntent:<note-text>*
- *exitKeyguardSecurely:<note-text>*
- *inKeyguardRestrictedInputMode:<note-text>*
- *newKeyguardLock:<note-text>*