Guion de Prácticas

Modelos de Computación
1º Grado en Ingeniería de Sistemas de Información
Universidad CEU San Pablo
Sergio Saugar García

Abril de 2021

Proyecto software: Web-Scraping

Entrega Obligatoria: SI
Fecha límite de entrega:
7 de Mayo de 2021 (Entrega INTERMEDIA)
28 de Mayo de 2021 (Entrega FINAL)
Peso: 30 % de la nota final de la asignatura

Esta práctica es de **entrega obligatoria**. Tenga en cuenta que:

- La sección 4 recoge instrucciones sobre qué elementos deben ser entregados y, también, instrucciones sobre el procedimiento a seguir.
- La fecha límite para la entrega de la práctica está establecida en este enunciado. No se aceptarán entregas a partir de dicha fecha.

1. Introducción

El objetivo principal de este proyecto de programación es ilustrar la aplicación práctica de los conceptos teóricos revisados en la asignatura. Concretamente, se centrará en las características propias de los analizadores léxicos (definidos mediante autómatas finitos o expresiones regulares). Estos analizadores son utilizados, fundamentalmente, como una de las primeras etapas en los procesos de compilación. En contraposición, también se trabajará con otros analizadores más avanzados que permiten examinar textos estructurados. Como hilo conductor de ambas aproximaciones, se utilizarán las técnicas de *Web-Scraping*.

A continuación, en la sección 2 se establece el contexto y los requisitos que debe cumplir el proyecto software. La sección 3 especifica cuestiones propias de la evaluación de la práctica. Posteriormente, la sección 4 **indica los elementos que deben entregarse y cómo debe realizarse el envío al profesor**. Finalmente, la sección 5 establece los mecanismos por los que se resolverán las dudas correspondientes a esta práctica.

2. Enunciado

Web-Scraping[1] es el término utilizado para distinguir aquellas técnicas que extraen datos de los sitios *web*. Habitualmente, las técnicas de *web-scraping* se implementan en programas conocidos como *bots* o *web crawlers*. Estos programas analizan las páginas web, recopilando la información para la que han sido diseñados y almacenándola, posteriormente, en bases de datos u hojas de cálculo para poder ser procesada.

Algunos de estos programas se utilizan, por ejemplo, para obtener y comparar precios de distintas fuentes (por ejemplo, comparar precios del mismo producto en *Amazon*, *El Corte Inglés* o *Fnac*), obtener los hiperenlaces contenidos en una página *web* (tal y como hacen los *bots* de *Google* para descubrir nuevos sitios), etcétera.

En este proyecto de programación, se implementará la lógica de un *bot* que simulará el comportamiento de *Google* cuando intenta descubrir nuevas páginas. Es decir, se utilizarán técnicas de *web-scraping* para obtener los hiperenlaces de una página web. También se implementará la lógica necesaria para obtener todas las imágenes contenidas en una página web.

Para la realización de la práctica, es necesario que:

- Descargue y configure el IDE *Netbeans*[2].
- Descargue el proyecto software que se encuentra comprimido en el *Campus Virtual*, denominado (gisi-modcomputacion.tgz).
- **Cree un repositorio privado en GitHub**. Clone el repositorio en local. Descomprima el proyecto en dicho repositorio y haga un *commit* inicial denominado: *"Inicialización del proyecto"*.
- Gestione los permisos del repositorio GitHub para agregar al profesor de la asignatura como "colaborador" (usuario: *ssaugar*).
- Tenga en cuenta que **el uso de un repositorio privado en GitHub para gestionar esta práctica es obligatorio** y servirá:
  - En primer lugar, para **poder realizar sesiones de tutorías *online* con el profesor** de la asignatura de una manera más eficaz.
  - En segundo lugar, **en la evaluación, se tendrá en cuenta todo el trabajo que quede reflejado en el repositorio**. Por ello se recomienda que haga *commits* a menudo cada vez que introduzca algún cambio importante en el proyecto (por ejemplo, programar una característica solicitada o corrección de un error). En cualquier caso, se aconseja realizar *commits* única y exclusivamente si el software es estable (al menos compila).
  - Mantenga sincronizado el repositorio remoto constantemente.

2.1. Parte 1: Utilización de un analizador léxico de páginas *web*

Una de las técnicas de *web-scraping* más utilizadas es el análisis (o *parseado*) del código *HTML* de una página web.

En esta parte de la práctica, el alumno hará uso de un analizador léxico. Estos analizadores reciben un fichero de texto y lo leen carácter a carácter; guiados por expresiones regulares (o autómatas finitos), convierten grupos de estos caracteres en unidades léxicas. Por ejemplo, en el

[1] https://en.wikipedia.org/wiki/Web_scraping
[2] *Netbeans*. https://netbeans.apache.org/download/index.html

caso del analizador léxico del lenguaje *Java*, cada vez que reconoce que el carácter 'I' es seguido por el caracter 'F', genera un token (o unidad léxica) "*TOKEN_IF*".

En un compilador, los tokens generados por los analizadores léxicos, sirven como entrada del analizador sintáctico que, en base a una gramática, verifica si esas unidades léxicas están bien organizadas sintácticamente. Por ejemplo, el analizador sintáctico de *Java*, al recibir el token "*TOKEN_IF*", determinará cuáles de los tokens que vienen después, forman parte de una expresión *If-then-else*, y comprobará que esa expresión está bien formada (esto es, que se ajusta a la gramática definida).

2.1.1. *JFlex*, un generador de analizadores léxicos

Actualmente, existen diversas herramientas que nos permiten crear analizadores léxicos de una manera sencilla. En este caso, se utilizará *JFlex*[3] por su idoneidad al integrarse con *Java*. Este generador de analizadores crea, a partir de un fichero *jflex* de configuración (en el que se delimitan las diferentes unidades léxicas mediante expresiones regulares), un *parser* basado en autómatas finitos.

En la ruta src/es/ceu/gisi/modcomp/webcrawler/jflex del proyecto, se puede encontrar el fichero lexico-html.jflex, que describe los lexemas de nuestro analizador de *HTML*. Antes de comenzar con la práctica deberá generar la clase *Java* que servirá de analizador. Para ello, en el mismo directorio, ejecute el programa contenido en la clase *ParserGenerator*. Esto creará una clase, denominada *HTMLParser*[4]. Inspeccione el fichero lexico-html.jflex (averigüe cómo se crean los tokens, para ello, haga uso del manual de *JFlex)*. Inspeccione también la clase *HTMLParser*.

2.1.2. El analizador *HTMLParser*

El analizador que utilizaremos reconoce unidades léxicas propias de las etiquetas *HTML*[5]. Así, devuelve como tokens: apertura de etiqueta ("<"), cierre de etiqueta (">"), igual ("="), slash ("/"), valores que aparecen entrecomillados y cadenas de caracteres que conforman palabras[6].

Siguiendo estas pautas, el analizador léxico al encontrarse, por ejemplo, la etiqueta *<HTML>*, generará los siguientes tokens:

- *Token_<*
- *Token_Palabra* (con valor: "html")
- *Token_>*

Será responsabilidad del programa que reciba esos tokens, interpretar que esos tokens, con esos valores y en ese orden, recrean el comienzo de una etiqueta *HTML*.

2.1.3. Tareas a realizar

Deberá completar la implementación de la clase *JFlexScraper* (y definir las clases que crea necesarias para crear un código de calidad) para que, haciendo uso del analizador léxico *HTMLParser*, sea capaz de analizar un documento *HTML* (**complejo y REAL**) y:

- Obtener todas las URLs a las que hacen referencia los hiperenlaces que tenga.

[3] *JFlex*. http://www.jflex.de/download.html

[4] Si se ha dado cuenta, al generar la clase, la salida del programa indica el número de estados del autómata finito no determinista construido (NFA) y cómo ha quedado después de su paso y minimización a determinista (DFA).

[5] Podrá averiguar más sobre etiquetas *HTML* en https://www.w3schools.com/tags/. En esa página también encontrará un manual de *HTML*.

[6] Los tipos de tokens que devuelve el analizador están definidos en: src/es/ceu/gisi/modcomp/webcrawler/jflex/lexico.

- Obtener todas las URLs de las diferentes imágenes que están incrustadas en la página.
- Señalar si el documento *HTML* está bien balanceado. Es decir, que todas las etiquetas *HTML* que se abren también se cierran y, además, se cierran en el orden adecuado.

Además, deberá escribir el programa principal de *WebCrawler* para que, utilizando la clase *JFlexScraper*, realice las siguientes tareas:

- Inicializar *JFlexScraper* con un fichero que contenga un documento *HTML* que se corresponda con una página REAL y suficientemente compleja (en caso de duda, consulte con el profesor).
- Obtener todas las *URLs* de los hiperenlaces que tenga.
- Obtener todas las *URLs* de las imágenes que estén incrustadas.
- Guardar ambas *URLs* en un fichero en el disco duro.
- Mostrar por pantalla si el documento *HTML* está bien balanceado.

Tenga en cuenta que el código fuente que genere deberá estar debidamente documentado (incluyendo *javadoc*) y estructurado y reflejará todas las buenas prácticas de programación que ha adquirido (encapsulación, reutilización de código, genericidad, etcétera). Serán aspectos que se valorarán en la evaluación de la solución.

En los tests de prueba ya definidos encontrará pequeñas referencias de uso de las distintas clases del proyecto. En relación a los tests de prueba deberá:

- Ampliar los tests de prueba para incluir los nuevos métodos que deba escribir y **todos aquellos que sean necesarios para asegurar que su implementación funciona correctamente**.

2.2. Parte 2: Análisis de la estructura *DOM* de páginas *Web*

Otra de las técnicas de *web-scraping* consiste en, a partir de una página *web*, obtener el árbol *DOM*[7] que la representa, y recorrerlo para obtener información acerca de los diferentes nodos y sus contenidos[8].

2.2.1. *Jsoup* un analizador de páginas *web*

En esta parte de la práctica se utilizará la librería *JSoup*[9]. Esta librería permite construir, desde la URL (o fichero) de un documento *HTML*, un árbol *DOM* que represente su contenido. Así, una vez obtenido el árbol, se ofrece una API[10] que permite recorrer los diferentes elementos *HTML*, realizar búsquedas directas, extraer información, etcétera. Para comprender cómo se usa esta librería, se recomienda la lectura de su *libro de recetas*[11].

[7] *Document Object Model*. https://en.wikipedia.org/wiki/Document_Object_Model
[8] Puede encontrar un tutorial sobre qué es el árbol *DOM* y cómo hacer referencia a su contenido en: https://www.w3schools.com/xml/dom_intro.asp
[9] *JSoup*. Página de descarga. https://jsoup.org/download
[10] *JSoup*. API. https://jsoup.org/apidocs/
[11] *JSoup*. Cookbook. https://jsoup.org/cookbook

2.2.2. Tareas a realizar

Deberá completar la implementación de la clase *JsoupScraper* (y definir las clases que crea necesarias para crear un código de calidad) para que, haciendo uso del árbol *DOM*, sea capaz de analizar un documento *HTML* y:

- Obtener todas las URLs a las que hacen referencia los hiperenlaces que tenga.
- Obtener todas las URLs de las imágenes que están incrustadas en el documento.
- Obtener estadísticas de uso de varias etiquetas *HTML*.

Además, deberá escribir el programa principal de *WebCrawler* para que, utilizando la clase *JsoupScraper*, realice las siguientes tareas:

- Inicializar *JsoupScraper* con una dirección *URL* que apunte a un documento *HTML* que se corresponda con una página REAL y suficientemente compleja (en caso de duda, consulte con el profesor).
- Obtener todas las *URLs* de los hiperenlaces del documento.
- Obtener todas las *URLs* de las imágenes incrustadas en el documento.
- Guardar ambas *URLs* en un fichero en el disco duro.
- Mostrar por pantalla las estadísticas de uso de las siguientes etiquetas *HTML*: *a*, *br*, *div*, *li*, *ul*, *p*, *span*, *table*, *td* y *tr*.

Tenga en cuenta que el código fuente que genere deberá estar debidamente documentado (incluyendo *javadoc*) y estructurado y reflejará todas las buenas prácticas de programación que ha adquirido (encapsulación, reutilización de código, genericidad, etcétera). Serán aspectos que se valorarán en la evaluación de la solución.

En los tests de prueba ya definidos encontrará pequeñas referencias de uso de las distintas clases del proyecto. En esta parte también se le solicita:

- Ampliar los tests de prueba para incluir los nuevos métodos que deba escribir y **todos aquellos que sean necesarios para asegurar que su implementación funciona correctamente**.

2.2.3. Tarea opcional

Como parte opcional de la práctica, que le permitirá obtener nota extra, se le propone realizar la siguiente tarea:

- Escoja una página web de su gusto que contenga datos de interés (resultados de futbol, resultados de sorteos de primitiva o similares, compras de productos, etcétera).
- Analice la estructura de la página e implemente los métodos adecuados, en la clase *JsoupScraper*, para extraer la información deseada y almacenarla en un fichero del disco duro.
- Incluya las líneas correspondientes en el programa principal de *WebCrawler* para ejecutar esas sentencias.
- Añada los test de prueba que considere oportunos.

La página a utilizar deberá tener una complejidad adecuada. **Consulte con el profesor antes de realizar esta parte opcional**.

3. Evaluación

Para superar la práctica, el alumno tendrá que realizar una defensa (respondiendo a preguntas formuladas por el profesor sobre funcionamiento, cuestiones de diseño, partes del código, etcétera).

Además de la defensa, la práctica se evaluará en relación a:

- **Organización, estructuración y calidad del código fuente** (encapsulación, genericidad, reutilización, etcétera).
- **Trabajo continuado y detallado que el alumno haya realizado a lo largo de la práctica**. Para verificar este apartado se revisarán todos los commits realizados en el repositorio de GitHub (**que es obligatorio utilizar para el seguimiento de la misma**). Se tendrá en cuenta: **la calidad de los commit**s y la **realización de manera progresiva** de la práctica. Es decir, el trabajo en el repositorio debe **reflejar** que el alumno ha estado trabajando en la práctica de manera continuada.
- **Funcionamiento** ajustado a los requisitos establecidos.
- **Amplitud y variedad de los test de pruebas** (que demuestren el correcto funcionamiento de todo el código).
- **Claridad del código y adecuación a las normas de estilo** del lenguaje de programación *Java*.
- **Trabajo realizado en la entrega intermedia** (ver apartado 4.1).
- **Memoria y calidad** de la misma (presencia, estructuración de la información, claridad en la exposición,...).

4. Entrega

Tenga en cuenta que este proyecto de programación tiene **dos fechas de entrega**. Por una parte, la primera es una entrega intermedia cuyo objetivo es valorar el trabajo realizado por el alumno hasta ese momento. Esta entrega *debería incluir* el trabajo descrito en el apartado 2.1. Por otra parte, la segunda entrega incluye **todo** el trabajo solicitado en este enunciado.

4.1. Entrega INTERMEDIA

La entrega intermedia se corresponde con el desarrollo de las especificaciones establecidas en el apartado 2.1.

Como resultado de esta parte del proyecto se deberá entregar un único fichero *zip* que incluya: una **pequeña memoria** y **TODA la carpeta del proyecto Netbeans** desarrollado (asegúrese de que contiene todo el código fuente, los *jar* usados, etcétera).

En el repositorio, marque esta entrega realizando un *commit* denominado *"Entrega intermedia"* (asegúrese de que el *commit* queda registrado).

La confección de la memoria y el envío del fichero se realizarán siguiendo estas instrucciones:

- La memoria incluirá una portada con el nombre del alumno y se ajustará a la siguiente estructura:
  - Análisis y descripción del proyecto:

    - Resumen de uso de *JFlex*.
    - Autómata que ilustre el funcionamiento del analizador programado.
    - Decisiones de diseño tomadas hasta el momento y cómo se han implementado.
  - Listado de todo el código fuente.
  - La memoria se presentará en **formato PDF**.

- El fichero se entregará a través de la actividad establecida a tal efecto en el *Campus Virtual* de la asignatura. No se admitirán entregas por otra vías.

- Recuerde que la **entrega de la práctica deberá realizarse dentro del período de tiempo indicado**.

4.2. Entrega FINAL

La entrega final se corresponde con el **desarrollo completo del proyecto**, tal y como se describe en este enunciado.

Como resultado de esta parte del proyecto se deberá entregar un único fichero *zip* que incluya: una **memoria completa** y **toda la carpeta del proyecto Netbeans** desarrollado (asegúrese de que contiene todo el código fuente, los *jar* usados, etcétera).

La confección de la memoria y el envío del fichero se realizarán siguiendo estas instrucciones:

- La memoria incluirá una portada con el nombre del alumno y se ajustará a la siguiente estructura:
  - Análisis y descripción del proyecto:
    - Introducción a la práctica.
    - Parte 1: JFlex
      - Resumen del uso que ha realizado de *JFlex*.
      - Autómata que ilustre el funcionamiento del analizador programado.
      - Decisiones de diseño tomadas en la práctica y cómo se han implementado.
      - Descripción de cada una de las pruebas de tests definidas: por qué ha realizado cada una de las pruebas (qué deseaba comprobar), por qué cree que el conjunto de pruebas demuestran que su software es correcto.
    - Parte 2: JSoup
      - Resumen del uso que ha realizado de *JSoup*.
      - Decisiones de diseño tomadas en la práctica y cómo se han implementado.
      - Descripción de cada una de las pruebas de tests definidas: por qué ha realizado cada una de las pruebas (qué deseaba comprobar), por qué cree que el conjunto de pruebas demuestran que su software es correcto.
  - Listado de todo el código fuente.
  - Conclusiones (**que incluirán una valoración del tiempo dedicado a la práctica**).
  - La memoria se presentará en **formato PDF**.

- El fichero se entregará a través de la actividad establecida a tal efecto en el *Campus Virtual* de la asignatura. No se admitirá la entrega por otras vías.

- Recuerde que la **entrega de la práctica deberá realizarse dentro del período de tiempo indicado**, en caso contrario, se evaluará con cero puntos.

5. Tutorías

Se recomienda que el alumno establezca reuniones periódicas con el profesor para asegurarse de que está evolucionando correctamente en el desarrollo de la práctica. Las consultas y dudas que surjan en el desarrollo de esta práctica, se resolverán durante las horas de tutorías establecidas. Para ello, es recomendable concertar, previamente, una cita con el profesor (por ejemplo, a través de correo electrónico: sergio.saugargarcia@ceu.es).