PROCESAMIENTO DE INFORMACIÓN EN APLICACIONES TELEMÁTICAS

Segundo Parcial.

31 de mayo de 2022

EJERCICIO 1. (3 PUNTOS)

A partir del fichero `catalogo.xml`, proporcionado en la práctica 3, se pretende saber, **usando expresiones regulares**, cuantos conceptos de primer nivel tienen subconceptos de tercer nivel. Para ello se ha de considerar el contenido de los atributos `id` de los elementos `<concept>`, que están en siguiente jerarquía:

`<catalog> -> <datsets> -> <dataset> -> <concepts> -> <concept>`

Por ejemplo, una de las líneas del fichero a analizar podría ser la siguiente:

<concept id="https://datos.madrid.es/egob/kos/entidadesYorganismos/administracionCentral/OficinasRegistro" />

NOTA: Fíjese bien en el espacio en blanco que hay al final antes del cierre del elemento

Se considera un concepto de primer nivel aquel que aparece tras https://datos.madrid.es/egob/kos/

Ejemplos:

Concepto de primer nivel de nombre *entidadesYorganismos*:

https://datos.madrid.es/egob/kos/entidadesYorganismos/

Concepto de segundo nivel:

https://datos.madrid.es/egob/kos/entidadesYorganismos/AdministracionCentral/

Concepto de tercer nivel:

https://datos.madrid.es/egob/kos/entidadesYorganismos/AdministracionCentral/OficinasRegistro/

Se pide:

Crear una aplicación java que contenga una clase llamada `NumeroConcepts.java` con las siguientes características:

- Dispondrá de un método con el prototipo:

  ```
  public static Map <String,Integer> cuentasConcepts (File catalogo)
  ```

  que, mediante el uso de expresiones regulares, devolverá un mapa que contenga los nombres de los conceptos de primer nivel que tengan, al menos, un concepto de tercer nivel, y el nº de conceptos de tercer nivel. Cada entrada del mapa tendrá como clave el nombre del concepto de primer nivel y el nº de conceptos de tercer nivel asociados.

- El método `main()` invocará al método `cuentasConcepts()` y posteriormente mostrará por pantalla el contenido del mapa.

La salida será por consola y debe ser algo similar a lo que se aprecia en la figura:

```
entidadesYorganismos: 207
actividades:          38
```

Notas:

- Observe que algunas url que tiene que analizar contienen la letra ñ.
- Se le proporciona un esqueleto de la clase `NumeroConcepts.java` donde el método `main()` recoge el argumento con el que se invoca a la aplicación (el fichero catalogo.xml), se encarga de verificar que el fichero existe, invoca al método `cuentasConcepts ()`, y muestra por la salida estándar el contenido del mapa. Se indica mediante un comentario *TODO* donde hay que meter el código que se evaluará en este ejercicio.
- No es obligatorio usar el código que se le da en la clase `NumeroConcept.java`. Puede hacerlo como desee siempre y cuando cumpla con las especificaciones del enunciado.
- El fichero catalogo.xml debería poder estar en cualquier sitio, por lo que al invocar a la aplicación se puede pasar el path absoluto o relativo junto con el nombre del fichero. Por tanto, no debe existir dentro del programa ninguna referencia al path donde debe estar el fichero.

Entrega

Este ejercicio se entregará en moodle en "Espacio de entrega del ejercicio 1". Únicamente debe entregar el fichero `NumeroConcepts.java`, que deberá tener entre comentarios, al inicio del fichero, su nombre, apellidos y DNI.

EJERCICIO 2. (7 PUNTOS)

El enunciado de este ejercicio se complementa con un documento de nombre *Documento de apoyo al ejercicio 2* que encontrará en moodle dentro del recurso *Material para el examen*. En este enunciado, cuando sea necesario, se le indicará el nº de página del citado documento donde puede ampliar la información.

En la práctica 4 realizó un parser JSON que extraía información de los ficheros .json indicados en el atributo id de cada `dataset` del fichero de salida XML generado en la práctica 3. De estos ficheros extrajo información de los 5 primeros objetos del array de nombre *@graph*. Una de las propiedades de los objetos de ese array tiene como nombre de clave *"@id"*, cuyo valor **no** extrajo en la práctica 4. El valor de esta propiedad es también un fichero .json al que se accede a través de una url. Estos ficheros JSON son los que va a usar en este ejercicio. En la página 2 del documento de apoyo puede ver uno de los ficheros JSON analizados en la práctica 4 donde se destacan las propiedades, con nombre de clave *"@id"* y su valor, que contiene la url de un fichero .json.

De estos nuevos ficheros .json debe extraer dos propiedades mediante un nuevo parser JSON que debe implementar. En la página 6 del documento de apoyo tiene un ejemplo de uno de estos ficheros JSON donde puede ver las propiedades que debe obtener.

Tras el análisis de los ficheros .json deberá generar un fichero XML válido, respecto al esquema del documento `graphs.xsd`, que se le proporciona en Moodle.

Para realizar de este ejercicio deberá partir de la aplicación que ha realizado en la práctica 5 (aunque también puede usar la práctica 4) que debe descargar de Moodle. A esta aplicación deberá añadir dos nuevas clases, de nombre `JSONGraphParser` y `SalidaXMLExamen`. Además, deberá modificar la clase del programa principal y la clase `JSONDatasetParser`.

A continuación, se le indican las funcionalidades que deben tener las nuevas clases o las que debe añadir a las clases existentes para obtener el objetivo final de generar un fichero XML válido respecto al esquema del documento `graphs.xsd`:

Clase JSONDatasetParser:

Recuperar los valores de las propiedades con nombre de clave "@id" para añadirlas al mapa *mDatasetConcepts* que se le propuso que usara en la práctica 4 para obtener los valores pertinentes del fichero .json analizado. En la página 3 del documento de apoyo tiene un ejemplo gráfico de la colección *mDatasetConcepts* que generó en la práctica 4 y el añadido que tiene que hacer para este ejercicio.

Clase principal:

1) Recoger un quinto argumento con el nombre del fichero XML que se generará respecto al esquema del documento `graphs.xsd`. Este argumento contendrá el path absoluto o relativo junto con el nombre del fichero. No debe existir dentro del programa ninguna referencia al path donde debería estar el fichero. No es necesario realizar ninguna comprobación sobre si su extensión es .xml.

2) Crear una colección de tipo lista con las urls que ha obtenido, de los valores de las propiedades con nombre de clave "@id", tras realizar el parser con la clase JSONDatasetParser. En la página 4 del documento de apoyo tiene un ejemplo de los valores que puede contener la lista que debe crear a partir de la colección *mDatasetConcepts*. En el caso de que no sea capaz de generar esta lista puede a partir de la colección

mDatasetConcepts y, para no quedarse atascado, puede meter "a mano" 3 valores de urls válidas y así continuar con el ejercicio.

3) Añadir el método `getGraphValues()` cuyo prototipo es:
```
private static Map<String, <Map<String,String>> getGraphValues (List<String> listaIDs)
```

   Este método recibe como parámetro la lista del paso anterior, que contine las urls de los ficheros .json que se deben analizar. Tras realizar este análisis se obtendrán los valores que se necesitan para generar un fichero XML válido respecto al esquema del documento `graphs.xsd`. Estos valores se almacenarán en una colección de tipo `Map<String, <Map<String,String>>`.

   En la página 5 del documento de apoyo puede ver la estructura de las colecciones que se usan en el método `getGraphValues()`.

   Este método deberá hacer uso de la nueva clase JSONGraphParser que se encargará de analizar los ficheros .json de la lista. Puesto que esta lista contiene varias urls a ficheros .json, se deberá usar la clase JSONGraphParser varias veces, una por cada fichero .json. Tal y como se le pidió en la práctica 4, para optimizar el tiempo de ejecución, el procesamiento de cada fichero .json se debe realizar por un hilo diferente, creando tantos hilos como núcleos tenga el ordenador que está usando. Y al igual que se le indicó en la práctica 4, tenga en cuenta que, para evitar problemas de concurrencia, los hilos deben usar colecciones *Thread Safe* cuando sean objetos compartidos. Al terminar la ejecución todos los hilos tendrán relleno el mapa de tipo `Map<String, <Map<String,String>>` con la información necesario para generar el XML.

4) Invocar al método `generar()` de la clase `SalidaXMLExamen` pasándole como parámetro el mapa devuelto por `getGraphValues()` para obtener un String que se escribirá en un fichero XML.

Clase JSONGraphParser:

Las instancias de estas clases serán los hilos creados y puestos en marcha desde el método `getGraphValues()` de la clase principal.

Esta clase es un parser JSON de ficheros como el que se muestra en la página 6 del documento de apoyo. Si abre uno de estos ficheros desde un navegador observará que no está correctamente formateado y le puede ser difícil ver su contenido. Por este motivo se le proporciona en moodle los ficheros de nombre `11673268-15-anos-miscelanea.json`, `11675246-25-mayo-dia-africa.json` y `11691936-12-junio.json`, correctamente formateados, para que los tome como ejemplo (esto no significa que sean los únicos ficheros que debe analizar).

De cada fichero .json debe obtener únicamente dos propiedades (las que están en amarillo en la página 6 del documento de apoyo). Estas propiedades debe añadirlas al mapa de tipo `Map<String, <Map<String,String>>` que comparten todos los hilos y el que, cuando finalicen todos los hilos, podrá devolver el método `getGraphValues()` de la clase principal. En la página 7 del documento de apoyo puede ver un ejemplo de este mapa una vez que se han analizado 3 ficheros .json y obtenidas las propiedades pertinentes.

Clase SalidaXMLExamen:

Esta clase, similar a la clase `GenerarXML` que realizó en las prácticas 3 y 4, se encargará de generar el fichero XML válido respecto al esquema del documento `graphs.xsd`.

Dispondrá de un método llamado `generar()` cuyo prototipo es:

```
public static String generar (Map<String, Map<String, String>> mGraph)
```

Este método recibe el mapa que devuelve el método `getGraphValues()` de la clase principal y genera el String que se usará en la clase principal para escribirlo en un fichero XML.

La finalidad de este método es generar un String que contenga un documento XML válido respecto al esquema del documento `graphs.xsd`

Material para realizar este ejercicio

En Moodle encontrará, dentro del recurso *Material para el examen*, los siguientes ficheros:

- Esqueletos de las clases `JSONGraphParser.java` y `SalidaXMLExamen.java`
- Documento esquema `graphs.xsd`
- Ficheros .json de ejemplo, correctamente formateados, como los que tiene que analizar en la clase `JSONGraphParser.java`
- Documento de apoyo al ejercicio 2.pdf

Entrega

Este ejercicio se entregará en moodle en "Espacio de entrega del ejercicio 2". Únicamente debe entregar los ficheros java de su aplicación. Cada fichero deberá tener entre comentarios, al inicio de este, su nombre, apellidos y DNI.

No debe meter los ficheros .java en un fichero zip. Debe subirlos todos los ficheros de forma individual