

Albero Genealogico

PROGETTO LINGUAGGI E COMPILATORI 2021/22

Francesco Chiocchi | Linguaggi e Compilatori | 20/06/2022

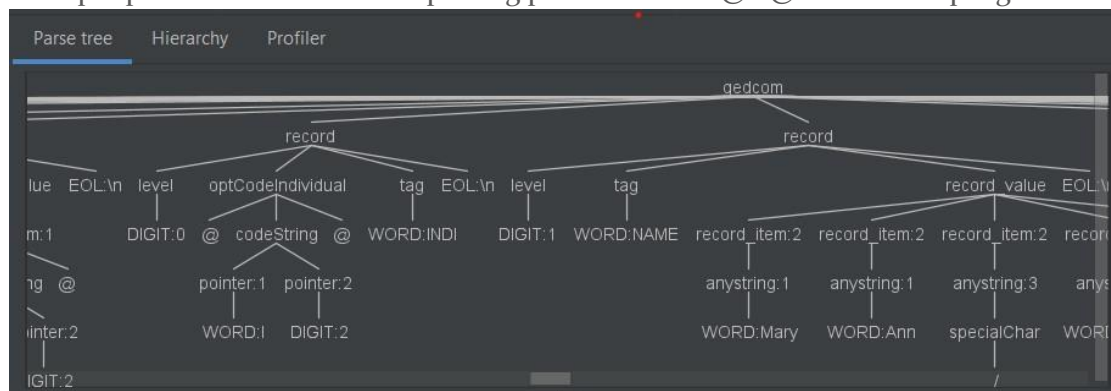
Grammatica

Usando il tool di IntelliJ per ANTLR4 ho costruito una grammatica generica dove ho immaginato che un file di tipo Gedcom è formato da un insieme di record, i quali hanno un livello, un tag opzionale rappresentante l'inizializzazione di un individuo o di una famiglia, un tag obbligatorio e il valore del record che è opzionale.

Infine, come detto all'inizio, un file di tipo Gedcom deve contenere anche un ultimo record di tipo richiesta dal quale poi partiranno le operazioni di calcolo degli antenati/discendenti di un certo individuo.

(Nella cartella condivisa metterò a disposizione l'immagine completa dell'albero di parsing, che ovviamente essendo molto grande non entra in Word).

Esempio parziale di un albero di parsing per l'individuo @I2@ nel file Sample.ged:



Implementazione

Innanzitutto, inizialmente avevo scelto di usare un visitor anziché un listener durante la visita dell'albero perché potevo ridirigere a mio piacimento le visite dei nodi. Poi mi sono reso conto che per il mio scopo non serviva quindi ho deciso di rimodificare la classe da cui fare overriding e ho usato quindi un semplice listener.

Il core dell'applicazione è proprio quando viene scoperta la grammatica, quindi quando entro nel metodo enterGedcom(ctx). Qui si inizializzano le mappe relative agli individui e alle famiglie presenti nella classe FamilyTree con un ciclo for visitando tutti i record presenti nel file. Vengono quindi aggiunti ad elements e families tutti i codici degli Individui/Famiglie, in cui il record il cui livello è zero e il tag obbligatorio è rispettivamente "INDI", per gli individui e "FAM" per le famiglie.

Contemporaneamente, per ogni record visitato, vengono impostate le caratteristiche di ogni individuo (nomi e cognomi se conosciuti, eventuali data e luogo di nascita, data e luogo di morte, luogo di sepoltura, puntatori al padre e alla madre se conosciuti) e ogni famiglia (eventuale data e luogo di Matrimonio, composizione di una famiglia).

Per ottenere poi tutti gli individui ho letto nell'ultima riga del file la richiesta, che può essere o di tipo ANCE o DESC, quando visito la regola enterRequest(ctx). Mi sono servito poi di un insieme generico Set<String> contenente tutti i codici degli individui risultanti dalla visita del file dopo aver preso la richiesta e chiamato i metodi getAncestorOf(Individual i) o getDescendantOf(Individual i), sempre a seconda del tipo di richiesta del file e poi li ho stampati a video quando la grammatica viene chiusa con exitGedcom(ctx).

Parti più importanti dell'enterGedcom(ctx):

```
for (GedcomParser.RecordContext r : l) {
    if (Integer.valueOf(r.level().getText()).equals(0) && r.tag().getText().equals("INDI")) {
        String codeIndividual = '@' + r.optCodeIndividual().codeString().getText() + '@';
        Individual individual = new Individual(codeIndividual);
        familyTree.addIndividual(individual);
        lastIndividual = individual;
    }
    setCharacteristicOfIndividual(r, lastIndividual);
    if (Integer.valueOf(r.level().getText()).equals(0) && r.tag().getText().equals("FAM")) {
        this.childs.clear();
        this.husb = null;
        this.wife = null;
        String codeFamily = '@' + r.optCodeIndividual().codeString().getText() + '@';
        AFamily aFamily = new AFamily(codeFamily);
        familyTree.addFamily(aFamily);
        lastFamily = aFamily;
    }
    if (Integer.valueOf(r.level().getText()).equals(1) && r.tag().getText().equals("HUSB")) {
        lastFamily.setHusb(r.record_value().record_item(0).getText());
        Individual husb = familyTree.getIndividual(lastFamily.getHusb());
        this.husb = husb;
    }
}
```

```
public Set<String> getAncestorsOf(String code) {
    if(!isPresent(code))
        throw new IllegalArgumentException("Il codice associato all'individuo non è presente");
    Set<String> s = new HashSet<>();
    Individual i0 = this.getIndividual(code);
    s.add(code);
    if (i0.getFather() != null)
        s.addAll(this.getAncestorsOf(i0.getFather().getCode()));
    if (i0.getMother() != null)
        s.addAll(this.getAncestorsOf(i0.getMother().getCode()));
    this.code = s;
    return s;
}

/** Restituisce l'insieme degli indici che sono discendenti di un individuo ...*/
public Set<String> getDescendantsOf(String code) {
    if(!isPresent(code))
        throw new IllegalArgumentException("Il codice associato all'individuo non è presente");
    Set<String> s = new HashSet<>();
    Individual i0 = this.getIndividual(code);
    s.add(code);
    for (Individual i : i0.getChilds()) {
        s.addAll(this.getDescendantsOf(i.getCode()));
    }
    this.code = s;
    return s;
}
```

Testing

Si è passato infine all'ultima fase, quella di testing, dove si allegano i vari screenshots dei file EsempioRossi.ged e Sample.ged. Inoltre, ho aggiunto un ulteriore file Gedcom riguardante la famiglia dei Sayan del manga “**Dragon Ball**”.

EsempioRossi.ged → Richiesta = o ANCE @I322382377075@

The screenshot shows an IDE with the following components:

- File Explorer (Left):** Displays the project structure for 'EsemplioRossi.ged'. It includes a 'gen' directory containing several classes: 'GedcomInterp', 'GedcomTokens', 'GedcomBaseListener', 'GedcomBaseVisitor', 'GedcomLexer', 'GedcomLexer.interp', 'GedcomLexer.tokens', 'GedcomListener', and 'GedcomParser'.
- Code Editor (Center):** Shows the 'GedcomBaseListener.java' file. It contains a GEDCOM file being parsed into a tree structure. The GEDCOM file content is:


```

      1 CHIL @I3223819343270
      0 @F60 FAM
      1 HUSB @I3223819343270
      1 WIFE @I3223819351510
      1 CHIL @I3223819350730
      0 @F70 FAM
      1 HUSB @I322381934860
      1 WIFE @I322381934920
      1 CHIL @I322381934320
      1 CHIL @I3223819349830
      0 TRLR
      0 ANCE @I3223823770750
      
```
- Console Window (Bottom):** Shows the output of the program. It includes the GEDCOM file content and the resulting tree structure:

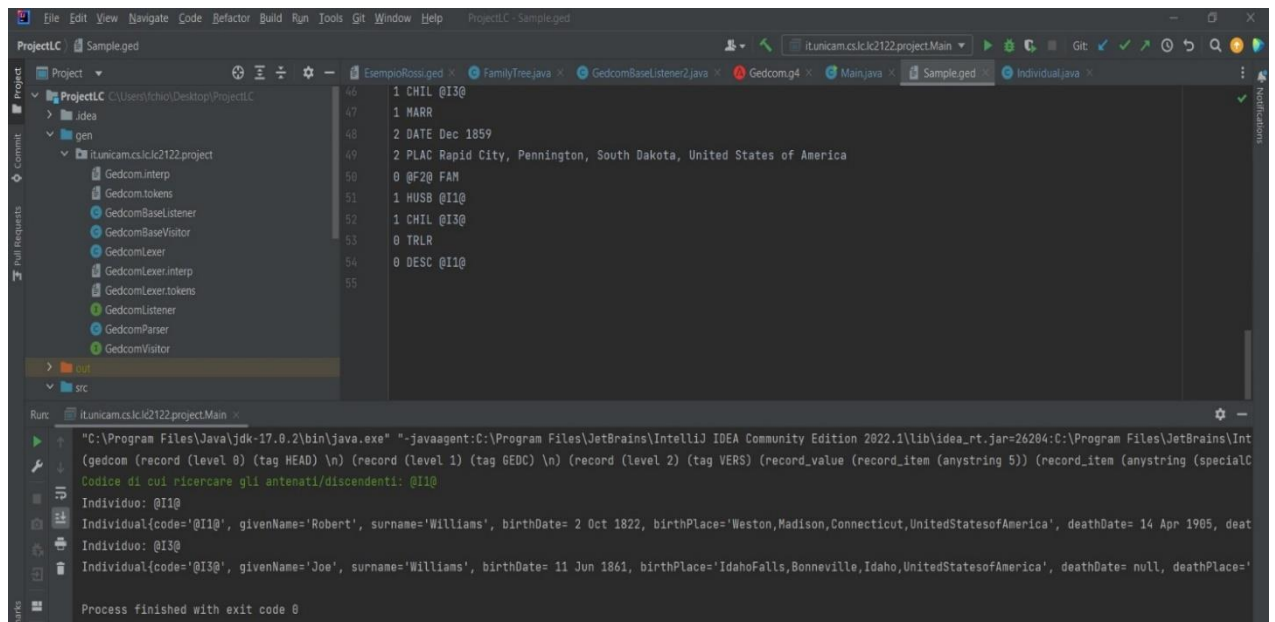

```

      "C:\Program Files\Java\jdk-17.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.1\lib\idea_rt.jar=26122:C:\Program Files\JetBrains\Int
      (gedcom (record (level 0) (tag HEAD) \n) (record (level 1) (tag DATE) (record_value (record_item (anysstring 18)) (record_item (anysstring May)) (record_item (anysstring 2022))) \n)
      Codice di cui ricercare gli antenati/discendenti: @I3223823770750
      Individuo: @I3223819347820
      Individual[code=@I3223819347820, givenName='Rosa', surname='Verdi', birthDate= 3 Mag 1983, birthPlace='Forlì, Emilia-Romagna, Italia', deathDate= 9 / 10, deathPlace='null', buryP
      Individuo: @I3223823768660
      Individual[code=@I3223823768660, givenName='Angela', surname='Gialli', birthDate= 4 Nov 1940, birthPlace='Pofi, Frosinone, Lazio, Italia', deathDate= null, deathPlace='null', bury
      Individuo: @I3223823770750
      Individual[code=@I3223823770750, givenName='Angela', surname='Bianchi', birthDate= 15 / 12, birthPlace='Frosinone, Lazio, Italia', deathDate= null, deathPlace='null', buryPlace='
      Individuo: @I3223819346690
      Individual[code=@I3223819346690, givenName='Marco', surname='Bianchi', birthDate= 5 Giu 1981, birthPlace='Bologna, Bologna, Emilia-Romagna, Italia', deathDate= 7 Ago 1974, deathPl
      Individuo: @I3223819354800
      Individual[code=@I3223819354800, givenName='Giovanni', surname='Bianchi', birthDate= 5 Dic 1929, birthPlace='Pisa, Pisa, Toscana, Italia', deathDate= null, deathPlace='null', bury
      
```

EsempioRossi.ged → Richiesta = o DESC @I322381934457@

The screenshot shows the IntelliJ IDEA IDE interface. The top bar contains the menu bar (File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, Git, Window, Help) and the title bar (Project.LC - EsemplioRossi.ged). The Project tool window on the left displays the project structure, including the 'itunicam.cs.lc2122.project' folder and its subfolders. The main editor window shows the 'EsemplioRossi.ged' file, which contains a list of names and IDs. The Run tool window at the bottom shows the output of the program, displaying a list of individuals with their details.

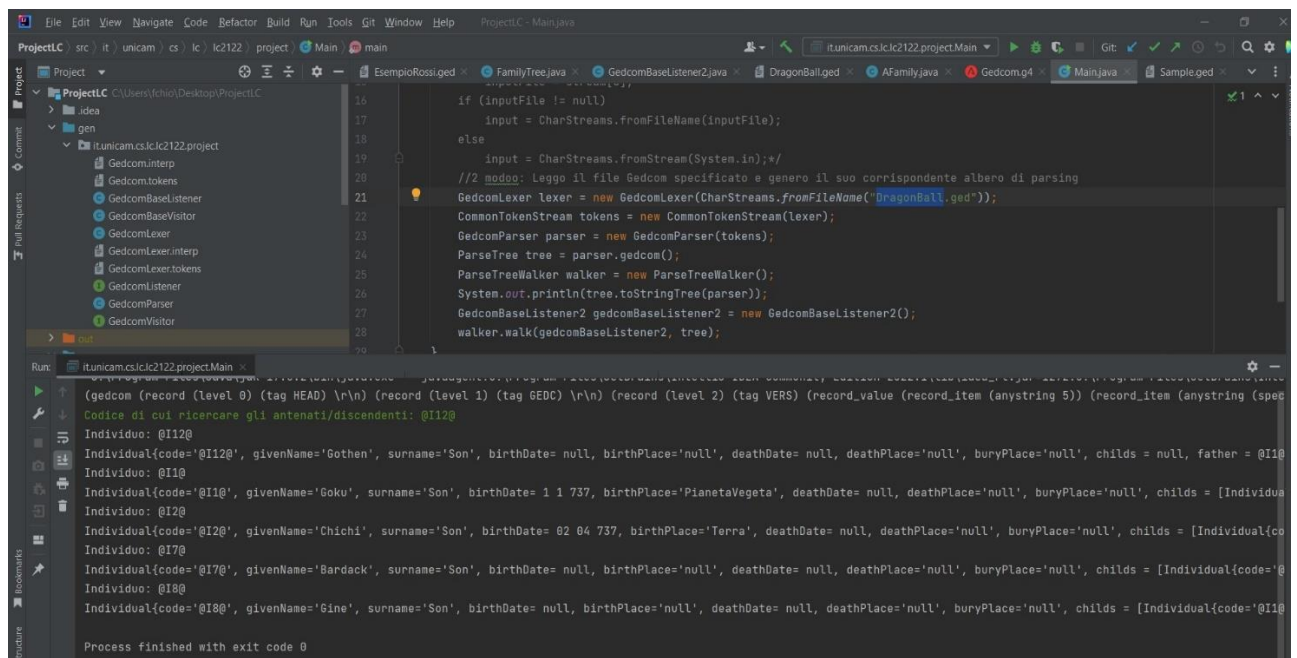
Sample.ged → Richiesta = o DESC @I1@



```
1 CHIL @I3@
1 MARR
2 DATE Dec 1859
2 PLAC Rapid City, Pennington, South Dakota, United States of America
0 @F2@ FAM
1 HUSB @I1@
1 CHIL @I3@
0 TRLR
0 DESC @I1@
```

```
"C:\Program Files\Java\jdk-17.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.1\lib\idea_rt.jar=26284:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.1\bin" -classpath C:\Users\ichio\Desktop\ProjectLC\out\classes;C:\Users\ichio\Desktop\ProjectLC\out\resources;C:\Users\ichio\Desktop\ProjectLC\lib\*.jar itunicam.cs.lc2122.project.Main
(gedcom (record (level 0) (tag HEAD) \n) (record (level 1) (tag GEDC) \n) (record (level 2) (tag VERS) (record_value (record_item (anystring 5)) (record_item (anystring (specialC
Codice di cui ricercare gli antenati/discendenti: @I1@
Individuo: @I1@
Individual{code='@I1@', givenName='Robert', surname='Williams', birthDate= 2 Oct 1822, birthPlace='Weston,Madison,Connecticut,UnitedStatesofAmerica', deathDate= 14 Apr 1905, deat
Individuo: @I3@
Individual{code='@I3@', givenName='Joe', surname='Williams', birthDate= 11 Jun 1861, birthPlace='IdahoFalls,Bonneville,Idaho,UnitedStatesofAmerica', deathDate= null, deathPlace='
Process finished with exit code 0
```

DragonBall.ged → o ANCE @I12@ (Gothen)



```
16 if (inputFile != null)
17     input = CharStreams.fromFileName(inputFile);
18 else
19     input = CharStreams.fromStream(System.in);
20 //2 modop: Leggo il file Gedcom specificato e genero il suo corrispondente albero di parsing
21 GedcomLexer lexer = new GedcomLexer(CharStreams.fromFileName("DragonBall.ged"));
22 CommonTokenStream tokens = new CommonTokenStream(lexer);
23 GedcomParser parser = new GedcomParser(tokens);
24 ParseTree tree = parser.gedcom();
25 ParseTreeWalker walker = new ParseTreeWalker();
26 System.out.println(tree.toStringTree(parser));
27 GedcomBaseListener2 gedcomBaseListener2 = new GedcomBaseListener2();
28 walker.walk(gedcomBaseListener2, tree);
```

```
(gedcom (record (level 0) (tag HEAD) \r\n) (record (level 1) (tag GEDC) \r\n) (record (level 2) (tag VERS) (record_value (record_item (anystring 5)) (record_item (anystring (spe
Codice di cui ricercare gli antenati/discendenti: @I12@
Individuo: @I12@
Individual{code='@I12@', givenName='Gothen', surname='Son', birthDate= null, birthPlace='null', deathDate= null, deathPlace='null', buryPlace='null', childs = null, father = @I1@
Individuo: @I1@
Individual{code='@I1@', givenName='Goku', surname='Son', birthDate= 1 1 737, birthPlace='PianetaVegeta', deathDate= null, deathPlace='null', buryPlace='null', childs = [Individual{
Individuo: @I2@
Individual{code='@I2@', givenName='Chichi', surname='Son', birthDate= 02 04 737, birthPlace='Terra', deathDate= null, deathPlace='null', buryPlace='null', childs = [Individual{co
Individuo: @I7@
Individual{code='@I7@', givenName='Bardack', surname='Son', birthDate= null, birthPlace='null', deathDate= null, deathPlace='null', buryPlace='null', childs = [Individual{code='@
Individuo: @I8@
Individual{code='@I8@', givenName='Gine', surname='Son', birthDate= null, birthPlace='null', deathDate= null, deathPlace='null', buryPlace='null', childs = [Individual{code='@I1@
Process finished with exit code 0
```

PROGETTO DISPONIBILE ANCHE IN: [ProjectLC](#)

*The
End*