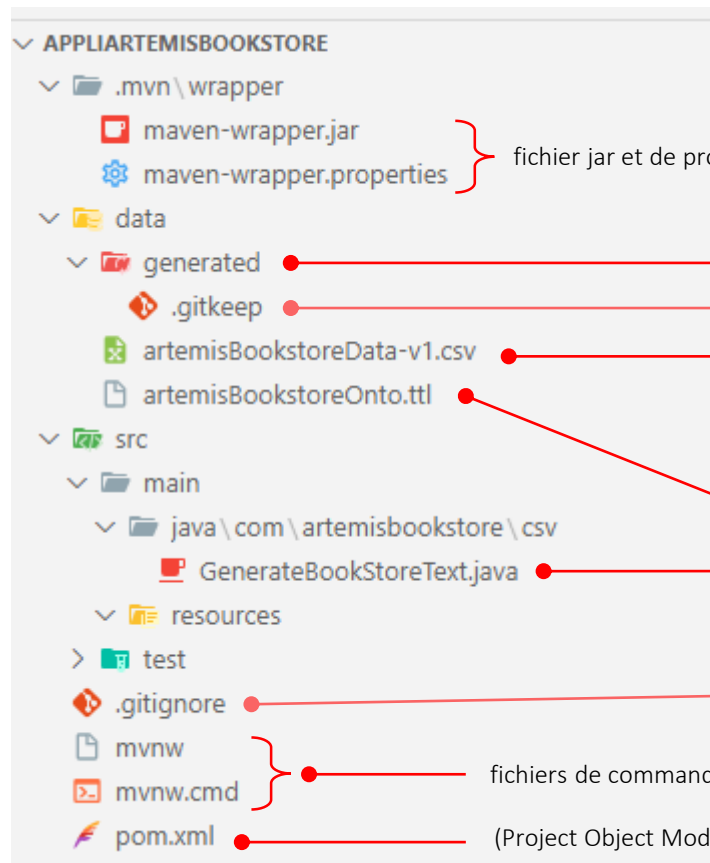


# Artemis BookStore

génération du graphe RDF à partir de données CSV

# Artemis BookStore : projet initial

- Structure du projet maven ArtemisBookStore ([ArtemisBookStore.zip](#))
  - projet Java maven utilisant maven wrapper (évite d'avoir à installer maven)
  - nécessite JDK 17 ou + (vérifier avec `javac -version` dans un terminal)



APPLIARTEMISBOOKSTORE

- `.mvn\wrapper`
  - `maven-wrapper.jar` et `maven-wrapper.properties` : fichier jar et de propriétés utilisés par maven wrapper
- `data`
  - `generated` : Répertoire où seront enregistrés les fichiers générés
  - `.gitkeep` : fichier vide pour forcer git à enregistrer le répertoire `generated` dans le repository (par défaut les répertoires vides ne le sont pas)
  - `artemisBookstoreData-v1.csv` : Le fichier de données CSV
    - ⚠ le séparateur utilisé est le `;` et non pas `,`
  - `artemisBookstoreOnto.ttl` : Le fichier contenant l'ontologie (en RDFS) définissant les classes et les propriétés utilisées pour décrire les données d'Artemis BookStore
- `src`
  - `main`
    - `java\com\artemisbookstore\csv`
      - `GenerateBookStoreText.java` : Le programme de conversion destiné à générer le graphe RDF en turtle. Situé dans le package `com.artemisbookstore.csv`
    - `resources`
  - `test`
    - `.gitignore` : définit les dossiers et fichiers à ignorer dans les commandes git
    - `mvnw` et `mvnw.cmd` : fichiers de commandes (unix et windows pour lancer maven wrapper)
      - ⚠ sous unix, penser à rendre exécutable le fichier `mvnw` (`chmod +x mvnw`)
    - `pom.xml` : (Project Object Model) contient des informations sur le projet et les configurations utilisées par Maven pour construire le projet.

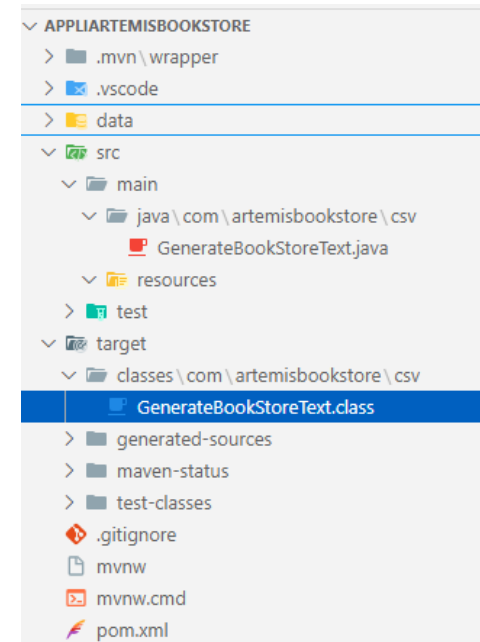
```
data > artemisBookstoreData-v1.csv > data
1  LastName;FirstName;BookTitle;PagesNb;Isbn;PublisherName
2  Abbey;Edward;Black Sun (Edward Abbey novel);159;978-1-55566-286-8;Simon & Schuster
3  Abbey;Edward;Fire on the Mountain (Abbey novel);211;0-8263-0457-5;Dial Press
4  Abbey;Edward;Hayduke Lives;352;ISBN 0-316-00411-1 (first edition, hardback);Little, Brown and Company
5  Abbey;Edward;Jonathan Livingston Seagull;40684;0-446-40684-1;Doubleday
```

# Artemis BookStore : projet initial

- compiler en ligne de commande :
  - depuis le répertoire racine du projet
  - `.\mvnw compile` (windows) ou `./mvnw compile` (linux)

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE  COMMENTS

PS P:\ENSEIGNEMENT\ServeursWEB\teaching\coursSW\tps\TP01_RDF\AppliArtemisBookStore> .\mvnw compile
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.artemisbookstore:artemisbookstore >-----
[INFO] Building artemisbookstore 1.0-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ artemisbookstore ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ artemisbookstore ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to P:\ENSEIGNEMENT\ServeursWEB\teaching\coursSW\tps\TP01_RDF\AppliArtemisBookStore\target\classes
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time:  0.983 s
[INFO] Finished at: 2024-10-16T10:13:40+02:00
[INFO]
PS P:\ENSEIGNEMENT\ServeursWEB\teaching\coursSW\tps\TP01_RDF\AppliArtemisBookStore> █
```



Un répertoire **target**  
contenant le code compilé  
est généré

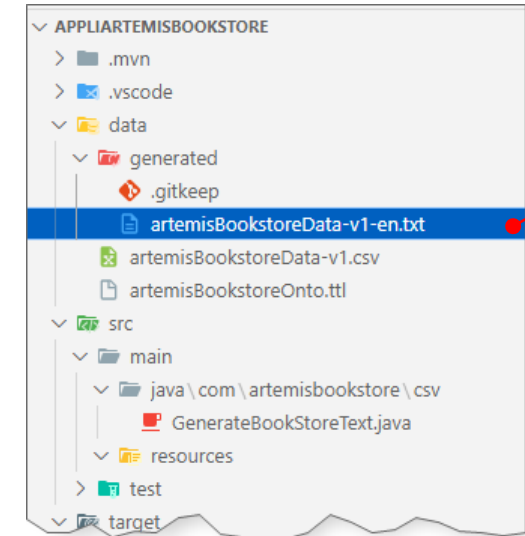
# Artemis BookStore : projet initial

- exécuter en ligne de commande :
  - depuis le répertoire racine du projet
  - `java -cp target\classes com.artemisbookstore.csv.GenerateBookStoreText`  
(séparateur de chemin \ sous window / sous linux)

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE  COMMENTS

PS P:\ENSEIGNEMENT\ServeursWEB\teaching\coursSW\tps\TP01_RDF\AppliArtemisBookStore> java -cp target\classes com.artemisbookstore.csv.GenerateBookStoreText

Text data has been generated
PS P:\ENSEIGNEMENT\ServeursWEB\teaching\coursSW\tps\TP01_RDF\AppliArtemisBookStore>
```



données CSV

```
data > artemisBookstoreData-v1.csv > data
1 LastName;FirstName;BookTitle;PagesNb;Isbn;PublisherName
2 Abbey;Edward;Black Sun (Edward Abbey novel);159;978-1-55566-286-8;Simon & Schuster
3 Abbey;Edward;Fire on the Mountain (Abbey novel);211;0-8263-0457-5;Dial Press
4 Abbey;Edward;Hayduke Lives;352;ISBN 0-316-00411-1 (first edition, hardback);Little, Brown and Company
5 Abbey;Edward;Jonathan Troy;374;1-131-40684-2;Dodd, Mead and Company
6 Abbey;Edward;The Monkey Wrench Gang;352;ISBN 0-397-01084-2 (hardback edition);Lippincott Williams & Wilkins
7 Abbey;Edward;The Brave Cowboy;277;0-8263-0448-6;Dodd, Mead and Company
8 Abbey;Edward;The Fool's Progress;485;0-8050-0921-3;Henry Holt and Company
9 Acito;Mark;How Paid for College: A Novel of Sex, the Friendship and Music;28;ISBN 0-8263-0448-6;Dodd, Mead and Company
```

fichier texte généré

```
data > generated > artemisBookstoreData-v1-en.txt
1 -----
2 Auteur [nom:Edward, prénom:Abbey]
3 Editeur [nom: Simon & Schuster]
4 Livre [titre: Black Sun (Edward Abbey novel), nbre pages: 159, isbn: 978-1-55566-286-8]
5 -----
6 Auteur [nom:Edward, prénom:Abbey]
7 Editeur [nom: Dial Press]
8 Livre [titre: Fire on the Mountain (Abbey novel), nbre pages: 211, isbn: 0-8263-0457-5]
9 -----
10 Auteur [nom:Edward, prénom:Abbey]
11 Editeur [nom: Little, Brown and Company]
12 Livre [titre: Hayduke Lives, nbre pages: 352, isbn: ISBN 0-316-00411-1 (first edition, hardback)]
13 -----
14 Auteur [nom:Edward, prénom:Abbey]
15 Editeur [nom: Dodd, Mead and Company]
16 Livre [titre: Jonathan Troy, nbre pages: 374, isbn: 1-131-40684-2]
17 -----
18 Auteur [nom:Edward, prénom:Abbey]
19 Editeur [nom: Lippincott Williams & Wilkins]
20 Livre [titre: The Monkey Wrench Gang, nbre pages: 352, isbn: ISBN 0-397-01084-2 (hardback edition)]
21 -----
```

# Artemis BookStore : projet initial

```
19
20 * @param csvFileName name of the input CSV file
21 * @param textFileName name of the output text file
22 *
23 * @throws java.io.IOException
24 */
25 public static void generateTurtle(String csvFileName, String textFileName) throws IOException {
26
27     try (BufferedReader br = Files.newBufferedReader(Paths.get(csvFileName));
28          BufferedWriter bw = Files.newBufferedWriter(Paths.get(textFileName))) {
29         // using Files.newBufferedReader and Files.newBufferedWriter enforce
30         // charEncodingto UTF-8.
31         // Files.newBufferedReader(Paths.get(turtleFileName)) is equivalent to
32         // BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(new
33         // FileOutputStream(turtleFileName), StandardCharsets.UTF_8));
34
35         String line;
36         br.readLine(); // to skip the column titles line
37         while ((line = br.readLine()) != null) {
38             // split the line in using the ';' (semicolon)
39             String[] lineTokens = line.split(";");
40
41             String authorFamilyName = lineTokens[0];
42             String authorGivenName = lineTokens[1];
43             String bookTitle = lineTokens[2];
44             int pagesNb = Integer.parseInt(lineTokens[3]);
45             String isbn = lineTokens[4];
46             String publisherName = lineTokens[5];
47
48             String outputline = ""
49                 .formatted(
50                     "Auteur [nom:%s, prénom:%s]",
51                     "Editeur [nom: %s]",
52                     "Livres [titre: %s, nbre pages: %d, isbn: %s]",
53                     ""
54                 )
55                 .formatted(authorGivenName, authorFamilyName, publisherName, bookTitle, pagesNb, isbn);
56             // System.out.println(outputline);
57             bw.write(outputline);
58         } // end of file
59     }
60
61     Run | Debug
62     public static void main(String[] args) throws IOException {
63         try {
64             generateTurtle("data/artemisBookstoreData-v1.csv", "data/generated/artemisBookstoreData-v1-en.txt");
65             System.out.println("\nText data has been generated");
66         } catch (IOException e) {
67             System.out.println("I/O Error while Generating Text Data");
68             e.printStackTrace();
69         }
70     }
71 }
```

ouverture du fichier en lecture (**BufferedReader br**) et en écriture (**BufferedWriter bw**)  
br et bw sont créés dans un try avec ressources (ils seront automatiquement fermés à la fin du bloc try)

boucle permettant de lire le fichier ligne par ligne

découpage de la ligne avec comme séparateur ';' en un tableau de **String** (**lineTokens**)

récupération des différentes valeurs pour la ligne courante

construction de la chaîne de sortie (on utilise une chaîne multilignes introduite dans JDK 15 délimitée par """, les formats %s et %d sont substitués par les valeurs des variables passées en paramètre de la méthode **formatted**)

écriture dans le fichier de sortie

programme principal

# Artemis BookStore : génération du graphe RDF

- Modifier le code de `GenerateBookStoreText.java` pour produire un graphe RDF au format Turtle

Les URIs pour identifier les ressources représentant un auteur, un éditeur et un livre seront de la forme `abr:uuid` où

- abr:** est le préfixe commun à toutes les ressources d'ArtemisBookStore, défini en Turtle par la directive  
`@prefix abr: <http://artemisBookstore.com/resource/>`
- uuid** est un identifiant unique universel ou UUID (Universally Unique Identifier). Les UUIDs sont utilisés pour identifier de manière unique des informations dans des systèmes informatiques. Ils sont généralement représentés par une chaîne de 36 caractères, comprenant des chiffres et des lettres, structurée en cinq groupes séparés par des tirets. Par exemple : 971fd38a-c35e-433c-af86-650c43c6c10b est l'UUID qui a été généré pour l'auteur Edward Abbey.

Les UUID sont conçus pour être uniques à travers le temps et l'espace, ce qui signifie qu'il est extrêmement improbable que deux UUID identiques soient générés indépendamment. Ils sont souvent utilisés dans les bases de données, les systèmes distribués, et les applications où des identifiants uniques sont nécessaires.

Pour générer un UUID en java vous pouvez utiliser la classe standard du JDK [java.util.UUID](#) comme le montre le programme ci-dessous.

```
import java.util.UUID;

public class UUIDExample {
    public static void main(String[] args) {
        UUID uuid = UUID.randomUUID();
        System.out.println("Generated UUID: " + uuid.toString());
    }
}
```

[javadoc](#) de `java.util.UUID`

artemisBookstoreData-v1.csv

| Last Name | First Name | Book Title                         | Pages Nb | Isbn           | Publisher Name                |
|-----------|------------|------------------------------------|----------|----------------|-------------------------------|
| Abbey     | Edward     | Black Sun (Edward Abbey novel)     | 159      | 978-1-55566-28 | Simon & Schuster              |
| Abbey     | Edward     | Fire on the Mountain (Abbey novel) | 211      | 0-8263-0457-5  | Dial Press                    |
| Abbey     | Edward     | Hayduke Lives                      | 352      | ISBN 0-316-004 | Little, Brown and Company     |
| Abbey     | Edward     | Jonathan Troy                      | 374      | 1-131-40684-2  | Dodd, Mead and Company        |
| Abbey     | Edward     | The Monkey Wrench Gang             | 352      | ISBN 0-39      | Lippincott Williams & Wilkins |

artemisBookstoreData-v1.ttl

```
data > generated > artemisBookstoreData-v1-en.ttl
1 @prefix abo: <http://artemisBookstore.com/ontology#> .
2 @prefix abr: <http://artemisBookstore.com/resource/> .
3 @prefix dcterms: <http://purl.org/dc/terms/> .
4 @prefix foaf: <http://xmlns.com/foaf/0.1/> .
5 @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
6
7 abr:971fd38a-c35e-433c-af86-650c43c6c10b a abo:Writer ;
8   foaf:givenName "Edward"@en ;
9   foaf:familyName "Abbey"@en ;
10  foaf:name "Edward Abbey"@en .
11
12 abr:60e677a9-4955-46c8-9aad-ddec1e8df747 a abo:Publisher ;
13   foaf:name "Simon & Schuster"@en .
14
15 abr:1ad53c26-4315-48d3-9d6e-50120360cf3a a abo:Book ;
16   dcterms:title "Black Sun (Edward Abbey novel)"@en ;
17   abo:author abr:971fd38a-c35e-433c-af86-650c43c6c10b ;
18   abo:isbn "978-1-55566-286-8" ;
19   abo:pages "159"^^xsd:int ;
20   dcterms:publisher abr:60e677a9-4955-46c8-9aad-ddec1e8df747 .
21
22 abr:c8aaaf3c-b299-452d-a2a0-ca13d4155056 a abo:Publisher ;
23   foaf:name "Dial Press"@en .
24
25 abr:3d31f6b6-ed6b-4c44-8a46-0b7a2b8b07da a abo:Book ;
26   dcterms:title "Fire on the Mountain (Abbey novel)"@en ;
27   abo:author abr:971fd38a-c35e-433c-af86-650c43c6c10b ;
28   abo:isbn "0-8263-0457-5" ;
29   abo:pages "211"^^xsd:int ;
30   dcterms:publisher abr:c8aaaf3c-b299-452d-a2a0-ca13d4155056 .
31
32 abr:8eae0e93-de0d-467c-a009-1371ff3485d1 a abo:Publisher ;
33   foaf:name "Little, Brown and Company"@en .
```

code RDF pour  
la 1<sup>ère</sup> ligne du  
fichier CSV

code RDF pour  
la 2<sup>ème</sup> ligne du  
fichier CSV

L'auteur n'est pas recréé  
on utilise l'UUID créé  
précédemment

UUID générés  
automatiquement  
pour avoir un  
identifiant unique  
pour chaque  
ressource

Auteur

Editeur

Livre

Editeur

Livre

# Artemis BookStore : génération du graphe RDF

```
18 // Maps to remember generated UUID for authors and publishers
19 private static final Map<String, UUID> authors = new HashMap<>();
20 private static final Map<String, UUID> publishers = new HashMap<>();
21
22 /**
23  *
24  * @param csvFileName name of the input CSV file
25  * @param turtleFileName name of the output turtle file
26  *
27  * @throws java.io.IOException
28  */
29 public static void generateTurtle(String csvFileName, String turtleFileName) throws IOException {
30
31     try {
32         BufferedReader br = Files.newBufferedReader(Paths.get(csvFileName));
33         BufferedWriter bw = Files.newBufferedWriter(Paths.get(turtleFileName)) {
34             // using Files.newBufferedReader and Files.newBufferedWriter enforce charEncoding
35             // to UTF-8
36             // Files.newBufferedReader(Paths.get(turtleFileName)) is equivalent to
37             // BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(new
38             // FileOutputStream(turtleFileName), StandardCharsets.UTF_8));
39
40             // write prefixes
41             bw.write("""
42                 @prefix abo: <http://artemisBookstore.com/ontology#> .
43                 @prefix abr: <http://artemisBookstore.com/resource/> .
44                 @prefix dcterms: <http://purl.org/dc/terms/> .
45                 @prefix foaf: <http://xmlns.com/foaf/0.1/> .
46                 @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
47
48                 """);
49
50             String line;
51             br.readLine(); // to skip the column titles line
52             while ((line = br.readLine()) != null) {
53                 // System.out.println(line);
54                 String[] lineTokens = line.split(",");
55
56                 String authorFamilyName = lineTokens[0];
57                 String authorGivenName = lineTokens[1];
58                 String bookTitle = lineTokens[2];
59                 int pagesNb = Integer.parseInt(lineTokens[3]);
60                 String isbn = lineTokens[4];
61                 String publisherName = lineTokens[5];
62                 String authorFamilyName = lineTokens[6];
```

Maps pour stocker les UUID déjà créés pour les ressources auteurs et éditeurs

écriture des préfixes dans le fichier de sortie

```
62 String publisherName = lineTokens[5];
63 String authorFullName = authorGivenName + " " + authorFamilyName;
64
65 UUID authorUUID = authors.get(authorFullName);
66 if (authorUUID == null) {
67     // this is the first occurrence of this author
68     // create triples describing him
69     authorUUID = UUID.randomUUID();
70     // store UUID in the map with authorFullName as key
71     authors.put(authorFullName, authorUUID);
72
73     String authorTemplateRDF = ""
74         abr:%s a abo:Writer;
75         foaf:givenName "%s"@en;
76         foaf:familyName "%s"@en;
77         foaf:name "%s"@en;
78
79     """;
80     bw.write(authorTemplateRDF.formatted(authorUUID, authorGivenName, authorFamilyName,
81         authorGivenName + " " + authorFamilyName));
82
83 }
84 UUID publisherUUID = publishers.get(publisherName);
85 if (publisherUUID == null) {
86     // this is the first occurrence of this publisher
87     // create triples describing it
88     publisherUUID = UUID.randomUUID();
89     // store UUID in the map with publisherName as key
90     publishers.put(publisherName, publisherUUID);
91
92     bw.write("""
93         abr:%s a abo:Publisher ;
94         foaf:name "%s"@en .
95
96         """).formatted(publisherUUID, publisherName));
97
98 }
99 UUID bookUUID = UUID.randomUUID();
100 bw.write("""
101     abr:%s a abo:Book ;
102     dcterms:title "%s"@en ;
103     abo:author abr:%s ;
104     abo:isbn "%s" ;
105     abo:pages "%d"^^xsd:int ;
106     dcterms:publisher abr:%s .
107
108     """).formatted(bookUUID, bookTitle, authorUUID, isbn, pagesNb, publisherUUID);
109
110 } // end of file
111 }
```

recherche dans la map des auteurs si un UUID a déjà été créé pour l'auteur (clé authorFullName)

c'est la première fois que l'auteur apparaît, on lui crée un UUID que l'on stocke dans la Map

écrite dans le fichier de sortie le code Turtle pour l'auteur

idem pour l'éditeur, on utilise la Map **publishers** pour ne pas créer deux fois une ressource pour un éditeur qui apparaîtrait sur plusieurs lignes

génération du Turtle pour le livre