Academia Tehnică Militară "Ferdinand I“
Facultatea de Sisteme Informatice şi Securitate Cibernetică

Paradigme de programare (în Java)

Lab 9/Curs 9- Servicii web REST

Col(r) Traian Nicula

Pentru acest laborator se creează un folder cu numele ***lab9*** în folder-ul cu numele studentului, unde vor fi create 2 proiecte astfel:

A. ***jackson*** - proiect Maven folosind ca artefact **archetype-quickstart-jdk8** care se salvează în folder-ul **lab9**. Pentru fiecare exercițiu se va crea câte o clasă de test (Exercise1 etc.) cu metoda statică *main*, precum și alte clase cerute de exerciții.
Adăugarea dependențelor ***Jackson*:**
   - se deschide fișierul ***pom.xml*** aparținând proiectului
   - se copiază sub tag-ul <dependencies> fragmentele xml de mai jos și se salvează fișierul

```
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-core</artifactId>
    <version>2.14.1</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-annotations</artifactId>
    <version>2.14.1</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.14.1</version>
</dependency>
```

   - fragmentele xml se găsesc pe site-ul ***MVN Repository*** la link-urile:

https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core
https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-annotations
https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind

Se rezolvă următoarele exerciții:

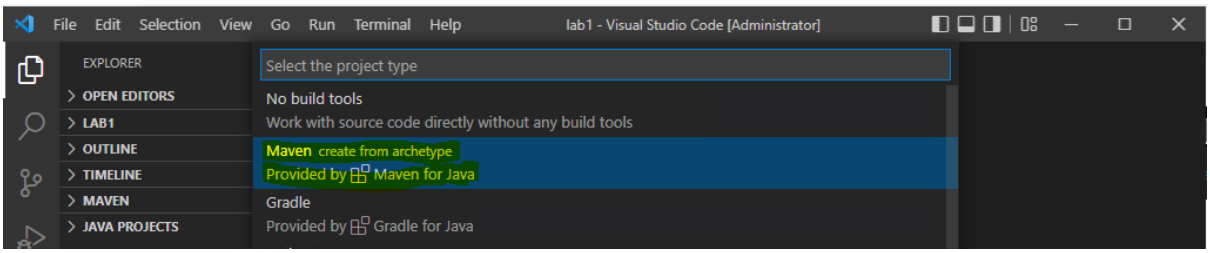
1. Scrieți un program Java care: (**2.5p**)
   - conține o clasă cu numele **Circle** având:
     a. câmpurile *int radius* și *String color*
     b. un constructor implicit și un constructor cu câmpurile *radius* și *color*
     c. metode getters, setters și toString
   - creează un String care conține o reprezentare JSON a unei instanțe a clasei Circle: ex. *{“radius”:5,”color”:”RED”}*
   - folosește metoda *readValue* a clasei *ObjectMapper* pentru a converti JSON la o instanță a clasei Circle
   - afișează rezultatul.
2. Scrieți un program Java care: (**2.5p**)
   - creează un fișier cu numele ***circle.json*** care conține o reprezentare JSON a unei instanțe a clasei Circle: ex. *{“radius”:5,”color”:”RED”}*
   - folosește metoda *getResourceAsStream* pentru a stoca fișierul într-un *InputStream*
   - folosește metoda *readValue* a clasei *ObjectMapper* pentru a converti *InputStream* la o instanță a clasei Circle
   - afișează rezultatul.
3. Scrieți un program Java care: (**2.5p**)
   - creează un fișier cu numele ***circleArray.json*** care conține o reprezentare JSON a unui vector cu 5 instanțe ale clasei Circle: ex. *[{“radius”:5,”color”:”RED”}, ...]*
   - folosește metoda *getResourceAsStream* pentru a stoca fișierul într-un *InputStream*
   - folosește metoda *readValue* a clasei *ObjectMapper* pentru a converti *InputStream* la o listă de instanțe a clasei *Circle* (se folosește *TypeReference* ca parametrul al doilea)
   - afișează rezultatul.
4. Scrieți un program Java care: (**2.5p**)
   - creează o instanță a clasei Circle cu valori pentru câmpurile *radius* și *color*
   - folosește metoda *writeValue* a clasei *ObjectMapper* pentru a scrie reprezentarea instanței Circle într-un fișier extern ***circleOut.json***.

Temă pentru acasă:

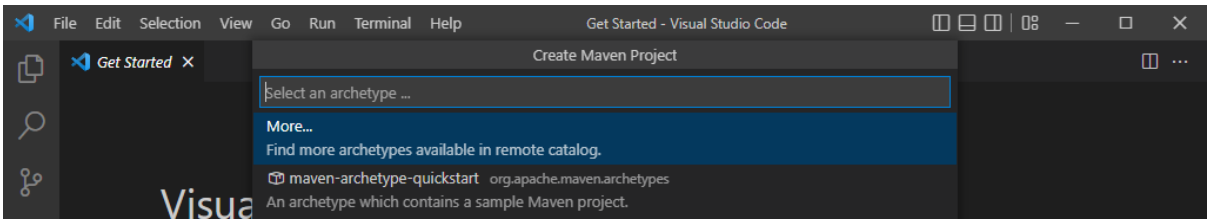
5. Scrieți un program Java care: (**1p**)
   - folosește clasa *ObjectNode* cu metodele *createObjectNode* și *put* pentru a crea programatic un obiect JSON cu următoarele chei: id, name, email, faculty, yearOfStudy și cu valori arbitrare
   - folosește metoda *writeValue* a clasei *ObjectMapper* pentru a scrie obiectul JSON într-un fișier extern ***studentOut.json***.
6. Scrieți un program Java care: (**1p**)
   - folosește metoda *getResourceAsStream* pentru a citi fișierul ***circle.json*** și a-l stoca într-un *InputStream*
   - creează un obiect *JsonNode*, prin parsare, folosind metoda *readTree* a clasei *ObjectMapper*
   - modifică culoarea cercului
   - afișează rezultatul.

B. ***restful*** - proiect Maven folosind ca artefact ***jersey-quickstart-webapp*** care se salvează în folder-ul **lab9**. Pentru fiecare exercițiu se ca crea câte o clasă resursă (Exercise1 etc.) cu metodele corespunzătoare, precum și alte clase cerute de exerciții. Arhetipul ***jersey-quickstart-webapp*** se găsește astfel:

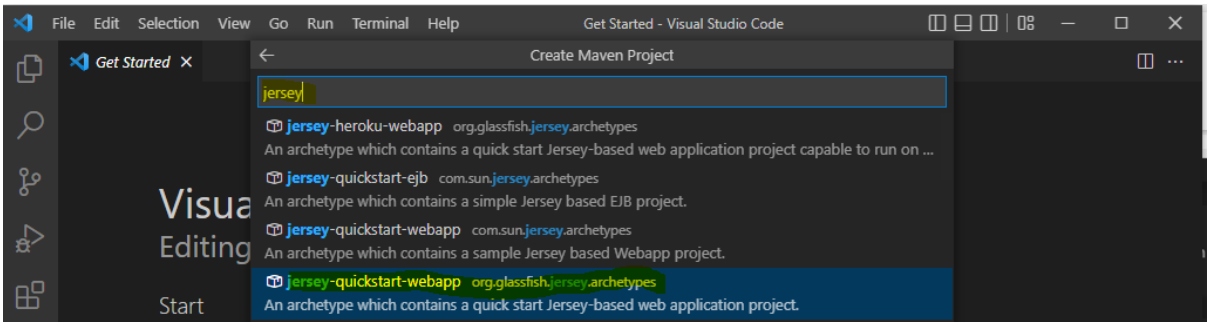
- Se urmează pașii anteriori până la alegerea tipului de proiect **Maven**



- Se alege **More ...** și se apasă **Enter**



- Se tastează ***jersey*** și se alege ***jersey-quickstart-webapp*** org.glassfish.jersey.archetypes



- Se alege versiunea **2.37**
- Se urmează pașii anteriori până la finalizarea generării și deschiderii proiectului
- Se deschide fișierul ***pom.xml***, se adaugă fragmentele xml și se salvează:

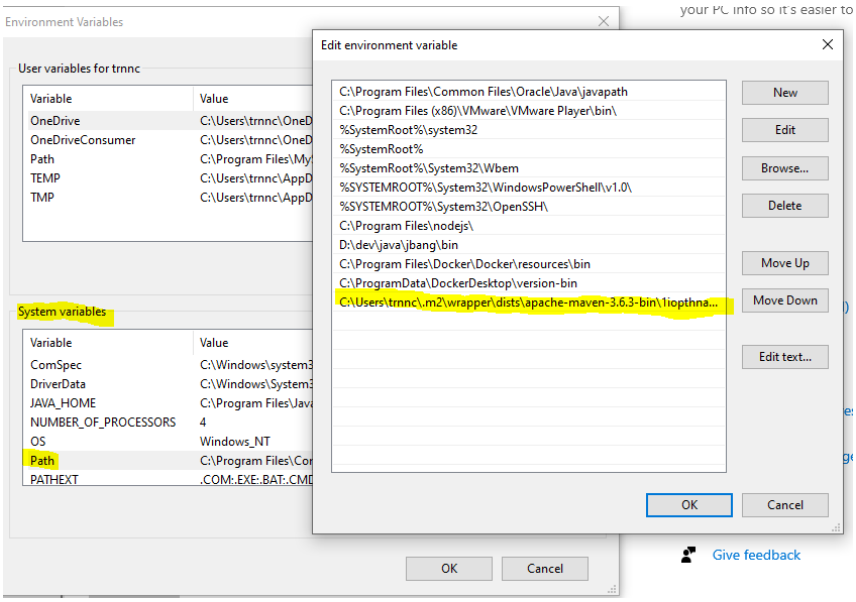
```
        <dependency>
            <groupId>org.glassfish.jersey.media</groupId>
            <artifactId>jersey-media-json-binding</artifactId>
        </dependency>
        <!--
https://mvnrepository.com/artifact/com.fasterxml.jackson.jaxrs/jackson-jaxrs-
json-provider -->
        <dependency>
            <groupId>com.fasterxml.jackson.jaxrs</groupId>
            <artifactId>jackson-jaxrs-json-provider</artifactId>
```

```
            <version>2.13.4</version>
        </dependency>
```

https://mvnrepository.com/artifact/com.fasterxml.jackson.jaxrs/jackson-jaxrs-json-provider/2.13.4

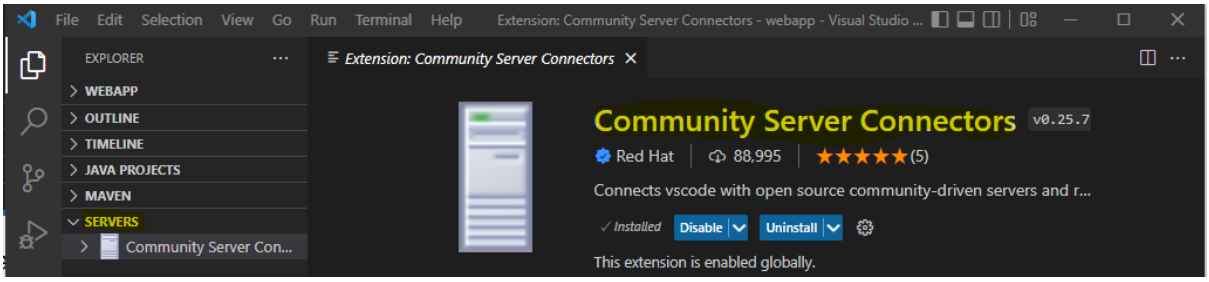
- Se adaugă calea către **MAVEN** (ultima versiune) la variabila de sistem **Path** (System variable):

C:\Users\trnnc\.m2\wrapper\dists\apache-maven-3.6.3-bin\1iopthnavndlasol9gbrbg6bf2\apache-maven-3.6.3\bin

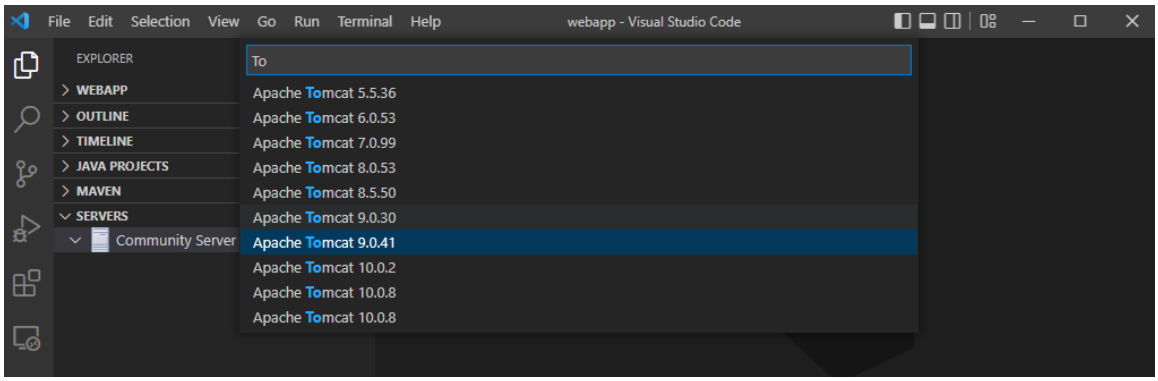


Pentru rularea compilarea și proiectului, este necesară ***instalarea Apache Tomcat***:

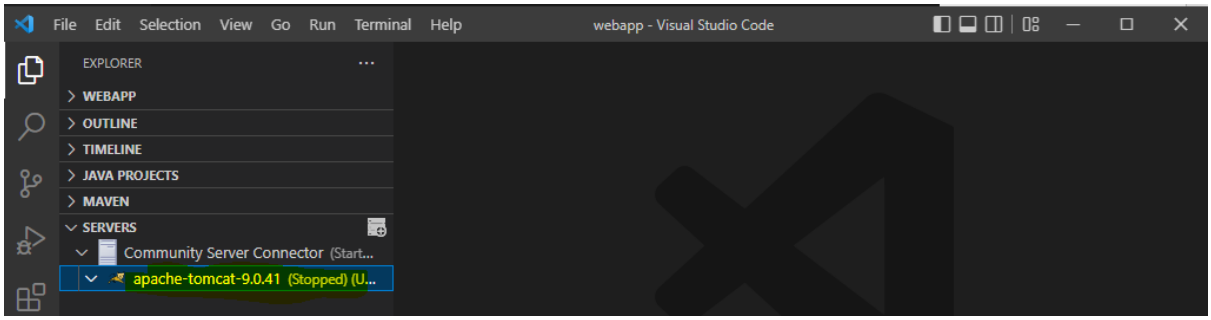
- Se instalează extensia VSC Community Server Connectors. După instalare apare SERVERS în panoul din stâng



- Se face clic dreapta pe nodul ***Community Server Connectors*** și se alege **Create New Server ...**
- Se apasă **Yes** pentru download și se alege serverul **Apache Tomcat 9.0.41**



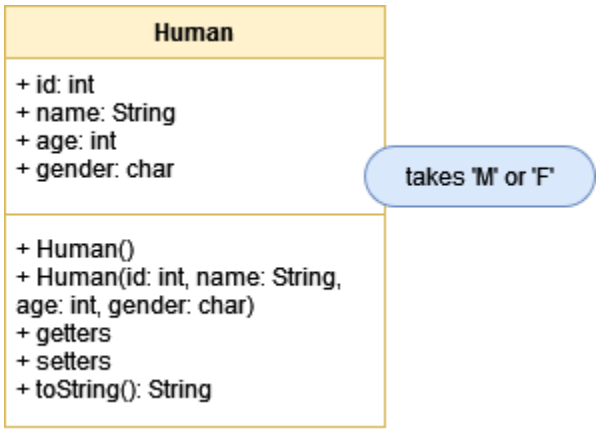
- Se afișează termenii de licențiere și se apasă butonul Continue și se confirmă termenii de licențiere
- Serverul Tomcat apare în panoul din stânga al VSC



- Se pornește serverul prin clic dreapta pe nod și alegerea opțiunii **Server start**
- Pentru a adăuga aplicația se alege opțiunea ***Add Deployment***. Se alege ***file*** și se selectează fișierul ***war*** (restful.war)
  Notă: *Fișierul se generează prin rularea în terminalul VCS a comenzii* ***mvn clean package***
  Fișierul war se găsește în ***\restful\target***
- Pentru a deschide în browser aplicația, prin clic dreapta pe nodul Tomcat se alege ***ServerActions ...***, apoi ***Show in Browser*** și URL-ul aplicației respectiv http://localhost:8080/restful/
- Se descarcă aplicația ***curl*** de la **https://curl.se/windows/**.
  Notă: *Se va utiliza pentru testarea serviciilor web REST*

Se rezolvă următoarele exerciții:

1. Scrieți un program Java care: (**5p**)
   - creează clasa ***Human*** conform diagramei UML de mai jos



- creează clasa ***Tools*** cu metoda statică *List<Human> getPeople()* care returnează o listă conținând 5 instanțe ale clasei
- creează clasa resursă (Exercise1 cu URI **/ex1)** având metoda *List<Human> getPeople()* care poate fi accesată cu metoda GET la URI **/people** și returnează lista oamenilor în format JSON
- generează pachetul (***mvn clean package)*** și testează rezultatul în browser.

2. Scrieți un program Java care: (**5p**)
   - creează clasa resursă (Exercise2 cu URI **/ex2)** având metoda *Human getPerson(int id)* care poate fi accesată cu metoda GET la URI **/people/{id}** și returnează reprezentarea persoanei în format JSON
   - generează pachetul (***mvn clean package)*** și testează rezultatul în browser (cu valori existente).

Temă pentru acasă:

3. Scrieți un program Java care: (**1p**)
   - creează clasa resursă (Exercise3 cu URI **/ex3)** având metoda *List<Human> addPeople(Human human)* care:
     a. poate fi accesat cu metoda POST la URI **/people**
     b. transmite în corpul cererii reprezentarea JSON a unei noi persoane
     c. creează local o copie a listei originale folosind Stream API (lista originală este nu este *thread safe*)
     d. adaugă persoana la listă și returnează lista în format JSON
   - generează pachetul (***mvn clean package)*** și testează rezultatul cu ***curl***.
4. Scrieți un program Java care: (**1p**)
   - creează clasa resursă (Exercise4 cu URI **/ex4)** având metoda *List<Human> deletePeople(int id)* care:
     a. poate fi accesată cu metoda DELETE la URI **/people/{id}**
     b. folosind Stream API (lista originală este nu este *thread safe*) elimină din listă persoana cu **id** specificat
     c. returnează lista în format JSON
   - generează pachetul (***mvn clean package)*** și testează rezultatul cu ***curl*** (cu valori existente).