תרגול בקורס בסיסי נתונים

תרגול 8

סמסטר ב' תשע"ט

RDF

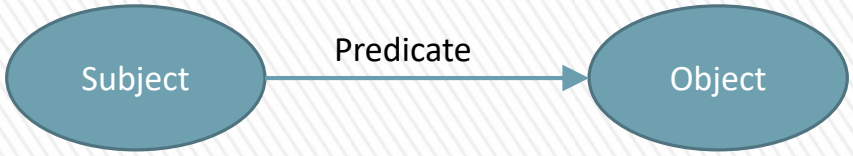
Resource Description Framework

RDF

- מסגרת המאפשרת תאור משאבים ברשת
- נועד לקריאה והבנה ע"י מחשבים ולא מיועד להצגה בפני אנשים
- כתוב ב-XML

RDF Triples

- מערך הנתונים מורכב מרשימה של שלשות מהצורה הבאה:



- כל הנתונים מוצגים רק ע"י השלשות הללו
- יש רק "טבלה" אחת, המכילה את כל השלשות הללו

RDF Example

« לדוגמא, מסמך פשוט המתאר את https://www.w3schools.com/rdf

```
<?xml version="1.0"?>

<RDF>
  <Description about="https://www.w3schools.com/rdf">
    <author>Jan Egil Refsnes</author>
    <homepage>https://www.w3schools.com</homepage>
  </Description>
</RDF>
```

Validator RDF

- מאפשר לבדוק את תקינות קבצי ה-RDF שלנו

http://www.w3.org/RDF/Validator/

RDF Example

« נניח שיש לנו את 2 הרשומות הבאות ברשימה של דיסקים:

Title	Artist	Country	Company	Price	Year
Empire Burlesque	Bob Dylan	USA	Columbia	10.90	1985
Hide your heart	Bonnie Tyler	UK	CBS Records	9.90	1988

```
<?xml version="1.0"?>

<rdf:RDF   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:cd="http://www.recshop.fake/cd#">

           <rdf:Description          rdf:about="http://www.recshop.fake/cd/Empire Burlesque">
                        <cd:artist>Bob Dylan</cd:artist>
                        <cd:country>USA</cd:country>
                        <cd:company>Columbia</cd:company>
                        <cd:price>10.90</cd:price>
                        <cd:year>1985</cd:year>
           </rdf:Description>

           <rdf:Description          rdf:about="http://www.recshop.fake/cd/Hide your heart">
                         <cd:artist>Bonnie Tyler</cd:artist>
                         <cd:country>UK</cd:country>
                         <cd:company>CBS Records</cd:company>
                         <cd:price>9.90</cd:price>
                         <cd:year>1988</cd:year>
           </rdf:Description>
           .
           .
</rdf:RDF>
```

The <rdf:RDF> Element

```
<?xml version="1.0"?>

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:cd="http://www.recshop.fake/cd#">

    ...Description goes here...

</rdf:RDF>
```

The <rdf:Description> Element

```
<?xml version="1.0"?>

<rdf:RDFxmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                    xmlns:cd="http://www.recshop.fake/cd#">
<rdf:Description
          rdf:about="http://www.recshop.fake/cd/EmpireBurlesque">

          <cd:artist>Bob Dylan</cd:artist>
          <cd:country>USA</cd:country>
          <cd:company>Columbia</cd:company>
          <cd:price>10.90</cd:price>
          <cd:year>1985</cd:year>
</rdf:Description>
....
</rdf:RDF>
```

Properties as Attributes

```xml
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">

<rdf:Description
rdf:about="http://www.recshop.fake/cd/Empire Burlesque"
cd:artist="Bob Dylan" cd:country="USA"
cd:company="Columbia" cd:price="10.90"
cd:year="1985" />

</rdf:RDF>
```

Properties as Resources

```
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">

<rdf:Description
rdf:about="http://www.recshop.fake/cd/Empire Burlesque">
  <cd:artist rdf:resource="http://www.recshop.fake/cd/dylan" />
  ...
  ...
</rdf:Description>

</rdf:RDF>
```

RDF Containers

- ניתן לתאר קבוצות של נתונים
- האמנטים המשמשים לתאר קבוצות הם:

  <Bag>, <Seq>, <Alt>

The <rdf:Bag> Element

```xml
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">

<rdf:Description
rdf:about="http://www.recshop.fake/cd/Beatles">
  <cd:artist>
    <rdf:Bag>
      <rdf:li>John</rdf:li>
      <rdf:li>Paul</rdf:li>
      <rdf:li>George</rdf:li>
      <rdf:li>Ringo</rdf:li>
    </rdf:Bag>
  </cd:artist>
</rdf:Description>

</rdf:RDF>
```

The <rdf:Seq> Element

```xml
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">

<rdf:Description
rdf:about="http://www.recshop.fake/cd/Beatles">
  <cd:artist>
    <rdf:Seq>
      <rdf:li>George</rdf:li>
      <rdf:li>John</rdf:li>
      <rdf:li>Paul</rdf:li>
      <rdf:li>Ringo</rdf:li>
    </rdf:Seq>
  </cd:artist>
</rdf:Description>

</rdf:RDF>
```

The <rdf:Alt> Element

```
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">

<rdf:Description
rdf:about="http://www.recshop.fake/cd/Beatles">
  <cd:format>
    <rdf:Alt>
      <rdf:li>CD</rdf:li>
      <rdf:li>Record</rdf:li>
      <rdf:li>Tape</rdf:li>
    </rdf:Alt>
  </cd:format>
</rdf:Description>

</rdf:RDF>
```

RDF Schema (RDFS)

- הרחבה ל-RDF.
- מאפשר הגדרה של מחלקות בדומה למחלקות בשפות תכנות מונחה עצמים.

RDFS Example

```xml
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base="http://www.animals.fake/animals#">

<rdf:Description rdf:ID="animal">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<rdf:Description rdf:ID="horse">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#animal"/>
</rdf:Description>

</rdf:RDF>
```

Example Abbreviated

```
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base="http://www.animals.fake/animals#">

<rdfs:Class rdf:ID="animal" />

<rdfs:Class rdf:ID="horse">
  <rdfs:subClassOf rdf:resource="#animal"/>
</rdfs:Class>

</rdf:RDF>
```

SPARQL:

- SPARQL היא שפת השאילתות על קבצי RDF.
- כל רכיב בשאילתא שהוא לא קבוע יתחיל ב'?'.
- כל שאילתא מסתיימת ב'.'
- כל שאילתא מורכבת משלשות:

Select ?????

Where Subject Predicate Object.

דוגמא:

ID	First_name	Last_name	age	email	gender
111	Alice	AAA	12	AAA@aaa	female
222	Bob	BBB	35	BBB@bbb	male

Subject	Predicate	Object
111	First_name	Alice
111	Last_name	AAA
111	Age	12
111	Email	AAA@aaa
111	Gender	Female
222	First_name	Bob
222	Last_name	BBB
222	Age	35
222	Email	BBB@bbb
222	gender	male

« נרצה לשלוף את כל הפרטים של בוב:

Select ?x where {

?Person first_name ”Bob”.

?Person ?pred ?x.

}

« שליפת כל כתובות האימייל במאגר:

Select ?E where ?person email ?E.

« שליפת שמות מלאים מהמאגר:

Select ?F ?L where {

?person First_name ?f.

?Person Last_name ?L.

}

« מה תציג השאילתא הבאה? –

Select ?ID ?A where {

?ID age ?A.

?ID gender “male”.

}

« תציג את תעודות הזהות והגילאים של כל הגברים.

Streams

- Stream הוא כלי בג'אווה המאפשר לנו להתייחס לאוספים כאל רצפים ומאפשר לנו לבצע פעולות וחישובים על האלמנטים הנמצאים באוסף.
- הפעולות יכולות להשפיע על כמות האלמנטים, סוג האלמנטים או על הסדר שלהם.
- דוגמאות-
  - filter >
  - map >
  - sorted >
  - match >
  - reduce ..... >

- אפשר להפוך ל-stream מטיפוסי נתונים שונים, כמו-
  - Array
  - List
  - i/o resources .....
- סוגי האופרציות ב-stream
  - intermediate
  - Terminal

- במהלך הדוגמאות הבאות נשתמש במערך

```
int[] arr = {1, 2, 45, 78, 3, 48, 23, 105, 5, 15};
```

- על מנת שנוכל לראות את התוצאות נוסיף בסוף הפקודה שלנו את הפקודה

```
forEach(System.out::println);
```

דוגמאות:
[1, 2, 45, 78, 3, 48, 23, 105, 5, 15]

- איך נחזיר את כל האיברים הקטנים מ10?
- Arrays.stream(arr).filter(s -> (s < 10))

- איך נחזיר את כל שאריות החלוקה ב10 של הערכים ברצף?
- Arrays.stream(arr).map(s->s%10)

- איך נחזיר את הרצף ממוין בסגר לקסיקוגרפי עולה?
- Arrays.stream(arr).sorted()

מה תדפיס לנו הפקודה הבאה?

```
» List<String> myList =
        Arrays.asList("a1", "a2", "b1", "c2", "c1");
        myList
        .stream()
        .filter(s -> s.startsWith("c"))
        .map(String::toUpperCase)
        .sorted()
        .forEach(System.out::println);
```

« יודפסו כל איברי הרצף המתחילים במחרוזת "c" עם אותיות גדולות במקום קטנות ובסדר לקסיקוגרפי עולה.

מה תדפיס לנו הפקודה הבאה?

» Arrays.asList("a1", "a2", "b1", "c2", "c1")
```
.stream()
.findFirst()
.ifPresent(System.out::println);
```

« יודפס האיבר הראשון ברצף.

אפשר ליצור גם stream מכלום!

```
Stream.of(1, 2, 45, 78, 3, 48, 23, 105, 5, 15).map(x -> x+1)
```

« יחזיר רצף שבו כל האיברים גדולים ב1 מהרצף שיצרנו.

collect

- Collect היא סוג של אופרציה שימושית מאד על streams.
- Collect הופכת את האלמנטים שנמצאים ב-stream לסוג אחר של נתונים, כמו למשל- list, set, map.
- ה-collect מקבל אובייקט שנקרא collector ולו יש כל מיני אופרציות כמו-
  - ToList
  - groupingBy
  - summarizingInt
  - Joining .....

:groupingBy

» Stream.of(1, 2, 45, 78, 3, 48, 23, 105, 5, 15)

```
.collect(Collectors.groupingBy(x->x%10));
```

» ,[15 ,5 ,105 ,45]=5 ,[23 ,3]=3 ,[2]=2 ,[1]=1}
{[48 ,78]=8

:summarizingInt

- Stream.of(1, 2, 45, 78, 3, 48, 23, 105, 5, 15)
  .collect(Collectors.summarizingInt((x->x)));

- IntSummaryStatistics{count=10, sum=325, min=1, average=32.500000, max=105}

:joining

- Stream.of("Hello", "world")
  .collect(Collectors.joining());
- Helloworld

- Stream.of("Hello", "world")
  .collect(Collectors.joining(" - "));
- Hello – world

- Stream.of("Hello", "world")
  .collect(Collectors.joining(" - ", "start " , " end"));
- Start Hello – world end

reduce

- פונקציית reduce לוקחת רצף של אלמנטים ומאחדת אותם לאיזשהיא תוצאה לפי מה שהוגדר.
- ניתן להשתמש בהגדרות קיימות כמו-
  - Sum >
  - Max >
  - Count .... >
- לדוגמא:

```
Stream.of(1,45, 6, 23, 8).reduce(Integer::sum);
```

Reduce

- ניתן לכתוב פונקציית reduce בעצמנו
- מבנה-

recude([identity], accumulator, [combiner])

> Accumulator: פונקציית למבדה עם 2 ערכי קלט- התוצאה עד עכשיו, האלמנט הנוכחי. הפונקציה מחזירה את התוצאה החלקית אחרי הוספת האלמנט הנוכחי.
> Combiner: פונקציית למבדה עם 2 ערכי קלט- 2 תוצאות חלקיות. הפונקציה מחזירה את התוצאה החלקית של שניהם יחד.
> Identity: אלמנט התחלתי שלא משנה את התוצאה (לדוגמא 0 במקרה של סכום, 1 במקרה של הכפלה)

- לדוגמא-

```
Stream.of(1,45, 6).reduce((x,y) -> x+y);
Stream.of(1,45, 6).reduce((x, y) -> x +" , " + y);
Stream.of(1,45, 6).reduce(0, (x,y) -> (x+1), (x,y)-> x+y)
```

דוגמא נוספת:

```
» students.stream().forEach(s ->{
        double average = s.getGrades().stream()
        .mapToDouble(g ->g.getValue()).average()
        .getAsDouble();
         System.out.println(
        s.getName() + "      average: " + average);
    });
```

דרך נוספת:

```
students.stream(). map(s -> {
        double average = s.getGrades().stream()
         .mapToDouble(g -> g.getValue())
         .average().getAsDouble();
        return new Pair(s, average);
      }).forEach(p -> {
         System.out.println(((Student)p.getKey()).get
         Name() + "'s average is " + p.getValue());
});
```

שאלת מבחן:

- מה יהיה הפלט של קטע הקוד הבא:

```
List<String> myList = Arrays.asList
("camel", "zebra", "you", "apple", "banana", "me");
myList.stream().map(a->a+a).filter(a->a.length()>=7)
        .sorted().forEach(System.out::println);
```

שאלת מבחן נוספת:

- כתבו פונקצית reduce בג'אוה stream שמקבלת רצף של מספרים ומחזירה את המכפלה של כל המספרים ההופכיים להם.

- תשובה:

```
Stream.of(1.0,2.0, 2.0, 3.0).reduce(1.0, (x,y) -> (x*(1/y)));
```

Parallel Streams

- במקרים בהם יש הרבה אלמנטים ואין חשיבות לסדר בינהם- ניתן להשתמש ב-()parallelStream במקום ב-()Streams
- כמובן שצריך לוודא שיש לנו threads פנויים...
- לדוגמא-

```
Arrays.asList(1,2,3).parallelStream()
    .reduce(2, (s1, s2) -> s1 * s2, (p, q) -> p + q)
```