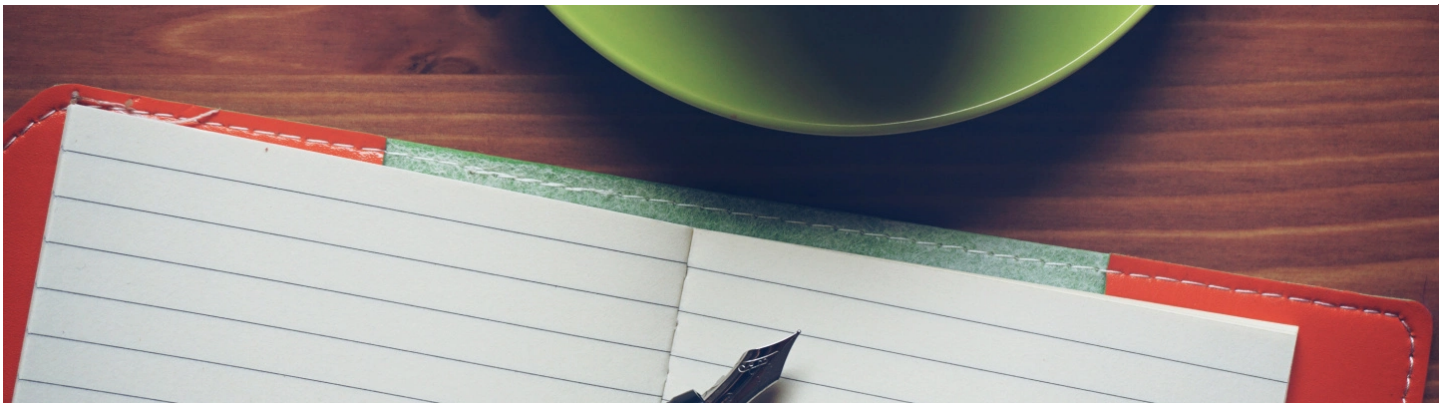
devs4j

EL MEJOR SITIO WEB SOBRE PROGRAMACIÓN EN ESPAÑOL.



Java 8 Streams – Transformaciones utilizando stream().map

HACE 3 DÍAS DEJA UN COMENTARIO

Continuado con los posts sobre Streams en este post explicaremos como transformar un objeto a otro utilizando la función map. Uno de los usos comunes de esto es transformar objetos a DTO's así que tomaremos este caso como ejemplo.

Definiendo el modelo

Antes de comenzar a utilizar streams definiremos dos clases a utilizar **Person** y **PersonDto,** las cuales serán la clase base de mi aplicación y la clase a la que quiero transformar mi objeto.

Person.java

```
1  public class Person {
2      private String firstName;
3      private String middleName;
4      private String lastName;
5      private String address;
6
7      public Person() {
```

```
8       }
9
10      public Person(String firstName, String
11          super();
12          this.firstName = firstName;
13          this.middleName = middleName;
14          this.lastName = lastName;
15          this.address = address;
16      }
17
18      public String getFirstName() {
19          return firstName;
20      }
21
22      public void setFirstName(String firstN
23          this.firstName = firstName;
24      }
25
26      public String getMiddleName() {
27          return middleName;
28      }
29
30      public void setMiddleName(String middl
31          this.middleName = middleName;
32      }
33
34      public String getLastName() {
35          return lastName;
36      }
37
38      public void setLastName(String lastNam
39          this.lastName = lastName;
40      }
41
42      public String getAddress() {
43          return address;
44      }
45
46      public void setAddress(String address)
47          this.address = address;
48      }
49
50  }
```

PersonDto.java

```
1   public class PersonDto {
2       private String firstName;
3       private String middleName;
4       private String lastName;
5       private String address;
6
7       public PersonDto() {
8       }
9
10      public PersonDto(String firstName, Str
11          super();
12          this.firstName = firstName;
13          this.middleName = middleName;
14          this.lastName = lastName;
```

```
        this.address = address;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstN
        this.firstName = firstName;
    }

    public String getMiddleName() {
        return middleName;
    }

    public void setMiddleName(String middl
        this.middleName = middleName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastNar
        this.lastName = lastName;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address)
        this.address = address;
    }
}
```

Las dos clases anteriores servirán como base para realizar mis transformaciones.

Creando objetos de ejemplo

De el mismo modo que en el post anterior, en este post utilizaremos Java-Faker para generar datos de ejemplo más información de como utilizarlo en el siguiente enlace Java-Faker (http://dius.github.io/java-faker/).

```
public static List getNames() {
    Faker faker = new Faker();
    List names = new ArrayList();
    for (int i = 0; i <= 1000; i++) {
        names.add(new Person(faker.name().f
                faker.address().fullAddress
    }
    return names;
```

```
9 }
```

El método anterior utiliza un Faker para generar datos de ejemplo y genera 1000 objetos Person con datos aleatorios.

Transformando objetos de la clase Person a PersonDto

Con el código anterior seremos capaces de generar 1000 objetos de tipo Person, el siguiente paso será transformarlos a objetos tipo PersonDto, veamos el siguiente ejemplo:

```
public static void main(String[] args) {
    List names = getNames();
    List peopleDtos = names.stream().map(person
        return new PersonDto(person.getFirstName
                person.getAddress());
    }).collect(Collectors.toList());
    for (PersonDto personDto : peopleDtos) {
        System.out.println("First name: " + pers
        System.out.println("Middle name: " + per
        System.out.println("Last name: " + perso
        System.out.println("Address: " + personD
        System.out.println("--------");
    }
}
```

El código anterior realiza lo siguiente:

1. Obtiene los 1000 objetos tipo Person a transformar
2. Utilizando Streams se ejecutará lo siguiente:
    1. Crear un stream
    2. Transformar un stream de objetos de tipo Person a uno de tipo PersonDto basado en el lambda que define la transformación
    3. Transformar la respuesta a una lista
3. Se imprimen los elementos pero ya con una lista tipo PersonDto

Salida:

```
1 --------
2 First name: Kenyatta
3 Middle name: Vallie
```

```
Last name: Runte
Address: Suite 750 049 Darrin Ford, East M
--------
First name: Sedrick
Middle name: Cara
Last name: Willms
Address: 36767 Lueilwitz Pines, Delphinebo
--------
First name: Luisa
Middle name: Merritt
Last name: Weissnat
Address: 00304 Cummings Viaduct, North Ela
--------
First name: Aniyah
Middle name: Deangelo
Last name: Keebler
Address: 37324 Prosacco Grove, North Shani
--------
First name: Edwardo
Middle name: General
Last name: Fadel
Address: 54550 Grimes Forge, Trantowboroug
--------
First name: Jensen
Middle name: Edd
Last name: Rutherford
Address: 31157 McGlynn Isle, Moiseston, HI
--------
1000 more...
```

Esto facilita la transformación de objetos de una clase a otra y es muy util al momento de escribir servicios web.

Para enterarte sobre futuros posts te recomendamos seguirnos en nuestras redes sociales: https://twitter.com/geeks_mx (https://twitter.com/geeks_mx) y https://www.facebook.com/geeksJavaMexico/ (https://www.facebook.com/geeksJavaMexico/).

Autor: Alejandro Agapito Bautista

Twitter: @raidentrance (https://geeksjavamexico.wordpress.com/mentions/raidentrance/)

Contacto:raidentrance@gmail.com