{MINGGU V}. SPRING BOOT DENGAN MONGODB

Oleh: TMP –2020/2021

Note: praktikum ini mengasumsikan anda telah menginstal dan mengkonfigurasi eclipse dengan Spring Boot plugin, serta terkoneksi dengan internet.

Sebelum anda memulai praktikum ini, pastikan segala tugas yang ada di minggu lalu sudah selesai. Hal ini dikarenakan tugas-tugas sebelumnya akan dipakai untuk di-integrasikan bersama dengan aplikasi lainnya.

Seperti yang telah disampaikan sebelumnya pada sesi online class, pengembangan aplikasi dengan menggunakan Spring Boot framework - bila menggunakan database MySQL - tidak ada bedanya dengan menggunakan database MongoDB. Hal ini dikarenakan Spring Boot memiliki driver/adapter yang "menyembunyikan" segala kerumitan yang terjadi di MongoDB. Oleh karena itu, anda benar-benar pure 100% tidak ada perubahan sama sekali di kode program Java anda.

Karena anda sekarang akan bekerja dengan MongoDB, alangkah lebih baik anda "berkenalan" lewat "mongo shell", agar lebih familiar dan fasih menggunakannya.

Getting familiar with Mongo Shell

Pastikan terlebih dahulu anda telah meng-configure mongo server lewat docker. Langkah 1- 3 berikut tidak perlu anda ikuti apabila mongo server telah aktif seperti pada gambar di bawah:

```
kuliah@ubuntu:~$ docker ps -a
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS
fb89c9036674        mongo                          "docker-entrypoint.s…"   4 hours ago         Up 4 hours
e24a6097fb03        mysql/mysql-server:latest      "/entrypoint.sh mysq…"   4 weeks ago         Up 3 weeks (healthy)
kuliah@ubuntu:~$
```

Figure 1. Status mongo server "Up"

Membuat user mongodb:

1. Downloading docker image

```
kuliah@ubuntu:~$ docker pull mongo
```

2. Buat virtual NIC

```
kuliah@ubuntu:~$ ifconfig ens33:2 192.168.164.223 netmask
255.255.255.0 up
```

Note: sesuaikan ip address dan subnet mask dengan environment VM anda. IP address yang di-set disini tidak boleh sama dengan ip address host dan juga VM.

3. Running mongodb container

```
kuliah@ubuntu:~$ docker run -d --name=mongodb_container \
-e MONGO_INITDB_ROOT_USERNAME=mongoadmin \
-e MONGO_INITDB_ROOT_PASSWORD=secret \
--publish 192.168.164.223:27017:27017 mongo
```

4. Executing mongodb container (masuk ke dalam container mongodb)

```
kuliah@ubuntu:~$ docker ps -a
```

```
-- lihat container id mongo, lalu copy id tersebut.

kuliah@ubuntu:~$ docker exec -it container_id bash
```

Membuat user mongodb:

5. Login ke mongodb server dengan user mongoadmin dengan authentikasi database **admin**.

```
kuliah@ubuntu:~$ mongo --username mongoadmin --password --host
192.168.164.223:27017 --authenticationDatabase admin
```

6. Buat user **admin** untuk database **admin**. Cara ini merupakan salah satu yang akan memungkinkan anda untuk membuat database lainnya, dan memiliki full-akses terhadap database tersebut.

```
> db.createUser({
      user: "admin",
      pwd: "secret",
      roles: [
                { role: "root", db: "admin" }
             ]
  });
```

7. Setelah anda berhasil membuat user **admin**, saatnya anda login dengan user tersebut. Namun sebelum login, anda harus logout dahulu dari user **mongoadmin** kemudian login kembali dengan user **admin**.

```
> exit;

kuliah@ubuntu:~$ mongo --username admin \
--password --host 192.168.164.223:27017
```

8. Buat database baru dengan nama “toped_db”.

```
> use toped_db;
```

9. Buat sebuah collection di database “toped_db” dengan nama “items”, lalu insert beberapa data seperti pada contoh di bawah ini.

```
> db.createCollection(“items”)
> db.items.insert({ item_name: "Real Me 7", stock: 15, price:
3600000, color: "Mist White", item_detail: {weight: 400,
condition: "new", insurance: "yes"}})
> db.items.insert({ item_name: "GUQI Juicer Cup Mini Electric
320ML Waterproof", stock: 23, sold: 36, seen: 7746, price: 90000,
item_detail: {weight: 600, condition: "new", insurance: "yes",
category: "household appliances"}})
> db.items.insert({ item_name: "Pioneer Earphone Extra Bass SE-
QL2T - Hitam", stock: 4, price: 299000, discount:60, color:
"Red", rating: 4.9, item_detail: {weight: 200, condition: "new",
insurance: "optional"}})
> db.items.insert({ item_name: "L636/12L lampu gantung laba laba
hias bisa atur teku dekor kafe retro", stock: 58, price: 750000,
```

```
discount:20, rating: 4.8, item_detail: {weight: 4000, condition:
"new", insurance: "optional"}})
> db.items.insert({ item_name: "Masker KN95 Respirator N95 Katup
Filter Udara 3D", stock: 116, price: 10900, sold: 3656,
seen:25825, rating: 4.8, item_detail: {weight: 50, condition:
"new", insurance: "optional", category: "healthcare"}})
> db.items.insert({ item_name: "MASKER KN95 N95 MASKER MEDIS 5
PLY ANTI VIRUS / BELI 10 GRATIS 1 - Hitam", stock: 241, price:
14500, sold: 3151, seen:28769, colour: "black", rating: 4.9,
item_detail: {weight: 50, condition: "new", insurance:
"optional", category: "healthcare"}})
> db.items.insert({ item_name: "Alat Fitness Treadmill iReborn
Paris", stock: 115, price: 6250000, accept_installment_payment:
"yes", lenght: 12, item_detail: {weight: 100000, condition:
"new", insurance: "yes"}})
> db.items.insert({ item_name: "cover Bean bag handle chair -
Biru", stock: 2, price: 250000, accept_installment_payment:
"yes", lenght: 6, color: "blue", item_detail: {weight: 1000,
condition: "new", insurance: "optional"}})
> db.items.insert({ item_name: "Canon EF 50mm f/1.8 STM", stock:
17, price: 1740000,sold: 32, seen: 4857, item_detail: {weight:
1000, condition: "new", insurance: "yes"}})
> db.items.insert({ item_name: "Tripod Handphone Tripod Kamera
3110 1 Meter Free Holder", stock: 77, price: 60000,sold: 232,
seen: 10907, item_detail: {weight: 650, condition: "new",
insurance: "optional", category: "camera accessories"}})
```

Tugas: gambarkanlah bentuk ERD dari data di atas. Beri nama file ERD yang telah anda kerjakan dengan "ERD_MONGODB_xxx.jpg", dimana **xxx** merupakan 3 digit terakhir NIM anda.

10. Buat kembali sebuah collection di database "toped_db" dengan nama "store", lalu insert beberapa data berikut:

```
> db.createCollection("store")
> db.store.insert({store_name: "Best_Com", location: "Jakarta
Barat", points: 476, reputation: "good", latest_online: 12})
> db.store.insert({store_name: "Aufaa_store18", location: "Kota
Bekasi", points: 128, reputation: "bad", latest_online: 3})
> db.store.insert({store_name: "Healthy_OK", location: "Bogor",
points: 674, reputation: "excellent", latest_online: 1})
> db.store.insert({store_name: "Allphone_jkt", location: "Jakarta
Utara", points: 100, reputation: "good", latest_online: 16})
> db.store.insert({store_name: "cozyme3", location: "Banten",
points: 153, reputation: "bad", latest_online: 60})
```

11. Untuk menampilkan semua data dari collection "items", dapat dilakukan dengan cara:

```
> db.items.find()
```

12. Untuk menampilkan data dengan kriteria tertentu dari collection “items” dapat dilakukan dengan cara:

```
> db.items.find({item_name: "Real Me 7"})
```

13. Untuk menampilkan data dengan kriteria tertentu dimana merupakan “dokumen anakan” dari collection “items” dapat dilakukan dengan cara:

```
> db.items.find({"item_detail.weight": 400})
```

14. Untuk meng-update data dari collection “items” dimana kriteria data yang ingin dicari adalah “item_name = “Real Me 7”” dan data yang ingin di update adalah “stock = 25”. Berikut caranya:

```
> db.items.update({item_name: "Real Me 7"}, { $set: {stock: 25} })
```

15. Anda juga dimungkinkan untuk menambah sebuah field baru pada sebuah dokumen di dalam collection “items” dengan kriteria “item_name = “Real Me 7””, dan nama field baru tersebut adalah “sold = 33”

```
> db.items.update({item_name: "Real Me 7"}, { $set: {sold: 33} })
```

*Note: ketika melakukan updating ataupun menambah field baru pada sebuah dokumen, disarankan menggunakan **ObjectId** dari dokumen yang dimaksud. Kali ini anda menggunakan **item_name** karena hanya sebagai contoh saja.*

16. Tugas: tambahlah field baru untuk setiap dokumen yang ada pada collection “items” dengan nama field “store_id”, dimana data yang diisikan haruslah sesuai pada table di bawah ini, simpan hasil kerjaan anda dengan nama UPDATE_SYNTAX_MONGODB_xxx.txt, dimana xxx merupakan 3 digits terakhir NIM anda.

Kriteria	Store Id
item_name: "Real Me 7"	store_name: "Allphone_jkt"
item_name: "GUQI Juicer Cup Mini Electric 320ML Waterproof"	store_name: "Aufaa_store18"
item_name: "Pioneer Earphone Extra Bass SE-QL2T - Hitam"	store_name: "Allphone_jkt"
item_name: "L636/12L lampu gantung laba laba hias bisa atur teku dekor kafe retro"	store_name: "cozyme3"
item_name: "Masker KN95 Respirator N95 Katup Filter Udara 3D"	store_name: "Healthy_OK"
item_name: "MASKER KN95 N95 MASKER MEDIS 5 PLY ANTI VIRUS / BELI 10 GRATIS 1 - Hitam"	store_name: "Healthy_OK"
item_name: "cover Bean bag handle chair - Biru"	store_name: "cozyme3"
item_name: "Canon EF 50mm f/1.8 STM"	store_name: "Best_Com"

item_name: "Tripod Handphone Tripod Kamera 3110 1 Meter Free Holder"	store_name: "Best_Com"
item_name: "Alat Fitness Treadmill iReborn Paris"	store_name: "Healthy_OK"

*Note: ketika melakukan updating ataupun menambah field baru pada sebuah dokumen, disarankan menggunakan **ObjectId** dari dokumen yang dimaksud. Kali ini anda menggunakan item_name karena hanya sebagai contoh saja.*

Working with Spring Boot - MongoDB

Berikut contoh yang sangat sederhana bagaimana menampilkan document dari sebuah collection, termasuk embedded document nya dengan menggunakan framework Spring Boot.

1. Buat project baru Spring Boot dengan dependency Thymeleaf, Spring Data MongoDB, Rest Repositories, dan Spring Web.
2. Buat 2 packages baru pada folder src/main/java dengan nama package del.ac.id.demo.controller dan del.ac.id.demo.jpa.
3. (Diasumsikan anda telah mengerjakan latihan di atas) pilih package del.ac.id.demo.jpa kemudian buat 2 kelas POJOs dengan nama Item.java dan ItemDetail.java. Berikut kode programnya.

File: ItemDetail.java

```java
package del.ac.id.demo.jpa;

import org.springframework.data.mongodb.core.mapping.Document;

@Document("item_detail")
public class ItemDetail {
    private double weight;
    private String condition;
    private String insurance;
    private String category;

    public ItemDetail() {}

    public ItemDetail(final double weight,
            final String condition,
            final String insurance,
            final String category) {
        this.weight = weight;
        this.condition = condition;
        this.insurance = insurance;
        this.category = category;
    }

    public void setWeight(double weight) { this.weight = weight; }
    public void setCondition(String condition) { this.condition = condition; }
    public void setInsurance(String insurance) { this.insurance = insurance; }
    public void setCategory(String category) { this.category = category; }

    public double getWeight() { return this.weight; }
    public String getCondition() { return this.condition; }
    public String getInsurance() { return this.insurance; }
    public String getCategory() { return this.category; }
}

```

File: Item.java

```java
package del.ac.id.demo.jpa;

import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.DBRef;
import org.springframework.data.mongodb.core.mapping.Document;
import org.springframework.data.mongodb.core.mapping.Field;

@Document("items")
public class Item {
    @Id
    String id;
    private String item_name, color, accept_installment_payment;
    private double stock, price, discount, rating;
    private int sold, seen, lenght;
    @DBRef
    @Field("item_detail")
    private ItemDetail itemDetail;

    public Item() {}

    public Item(final String item_name,
            final String color,
            final double stock,
            final double price,
            final double discount,
            final double rating,
            final int sold,
            final int seen,
            final int lenght,
            final ItemDetail itemDetail) {
        this.item_name = item_name;
        this.color = color;
        this.lenght = lenght;
        this.stock = stock;
        this.price = price;
        this.discount = discount;
        this.rating = rating;
        this.sold = sold;
        this.seen = seen;
        this.lenght = lenght;
        this.itemDetail = itemDetail;
    }

```

```
    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public void setItem_name(String item_name) { this.item_name = item_name; }
    public void setColor(String color) { this.color = color; }
    public void setAccept_installment_payment(String accept_installment_payment)
    { this.accept_installment_payment = accept_installment_payment; }
    public void setStock(double stock) { this.stock = stock; }
    public void setPrice(double price) { this.price = price; }
    public void setDiscount(double discount) { this.discount = discount; }
    public void setRating(double rating) { this.rating = rating; }
    public void setSold(int sold) { this.sold = sold; }
    public void setSeen(int seen) { this.seen = seen; }
    public void setLenght(int lenght) { this.lenght = lenght; }
    public void setItemDetail(ItemDetail itemDetail) { this.itemDetail = itemDetail; }

    public String getItem_name() { return this.item_name; }
    public String getColor() { return this.color; }
    public double getStock() { return this.stock; }
    public double getPrice() { return this.price; }
    public double getDiscount() { return this.discount; }
    public double getRating( ) { return this.rating; }
    public int getSold( ) { return this.sold; }
    public int getSeen() { return this.seen; }
    public int getLenght() {return this.lenght;}
    public ItemDetail getItemDetail() { return this.itemDetail; }
}
```

4. Kemudian buatlah service DAO untuk kelas POJO Item.java, letakan pada package del.ac.id.demo.jpa.

File: ItemRepository.java

```
package del.ac.id.demo.jpa;

import org.springframework.data.mongodb.repository.MongoRepository;

public interface ItemRepository extends MongoRepository<Item, String> {
}
```

5. Kemudian buatlah controller untuk POJO Item.java, dan letakan pada package del.ac.id.demo.controller

File: ItemController.java

```java
package del.ac.id.demo.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.ModelAndView;

import del.ac.id.demo.jpa.Item;
import del.ac.id.demo.jpa.ItemRepository;


@RestController
public class ItemController {
    @Autowired ItemRepository itemRepository;

    @RequestMapping("/item")
    public ModelAndView item() {
        List<Item> items = itemRepository.findAll();

        ModelAndView mv = new ModelAndView("item");
        mv.addObject("items", items);

        return mv;
    }
}
```

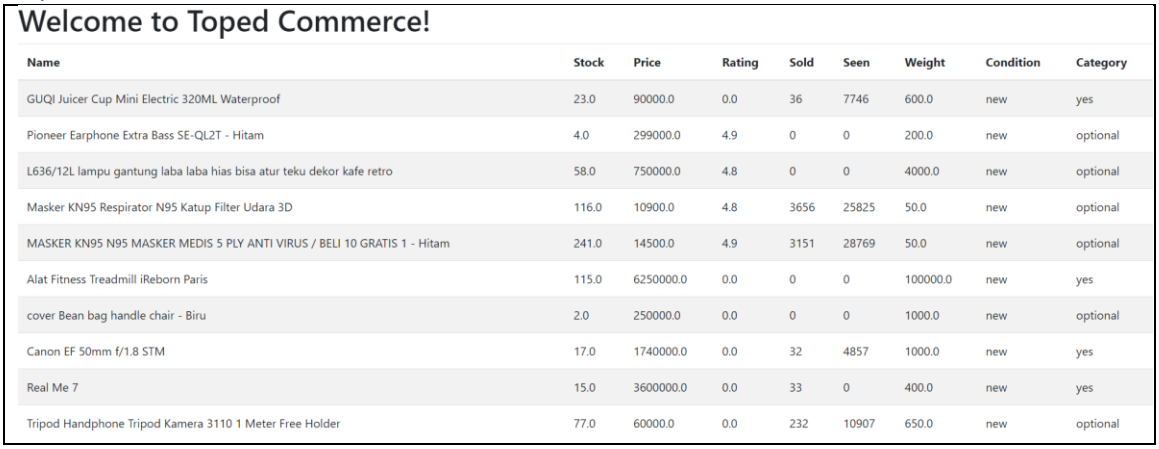
6. Terakhir buat halaman web dengan nama item.html dan letakan pada folder src/main/resources/templates

File: item.html

```html
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<head>
	<meta charset="utf-8"/>
	<link
href="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
rel="stylesheet" id="bootstrap-css">
<script
src="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
<script
src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<!------ Include the above in your HEAD tag ---------->

<link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.0.8/css/all.css">
</head>

<body>
	<h1>Welcome to Toped Commerce!</h1>
	<table class="table table-striped table-responsive-md">
		<thead>
			<tr>
				<th>Name</th>
```

```
                        <th>Stock</th>
                        <th>Price</th>
                        <th>Rating</th>
                        <th>Sold</th>
                        <th>Seen</th>
                        <th>Weight</th>
                        <th>Condition</th>
                        <th>Category</th>
                    </tr>
                </thead>
                <tbody>
                    <tr th:each="item : ${items}">
                        <td th:text="${item.item_name}"></td>
                        <td th:text="${item.stock}"></td>
                        <td th:text="${item.price}"></td>
                        <td th:text="${item.rating}"></td>
                        <td th:text="${item.sold}"></td>
                        <td th:text="${item.seen}"></td>
                        <td th:text="${item.itemDetail.weight}"></td>
                        <td th:text="${item.itemDetail.condition}"></td>
                        <td th:text="${item.itemDetail.insurance}"></td>
                    </tr>
                </tbody>
            </table>
    </body>
    </html>
```

7. Jalankan aplikasi web sederhana anda dengan cara mengakses URL http://localhost:8080/item

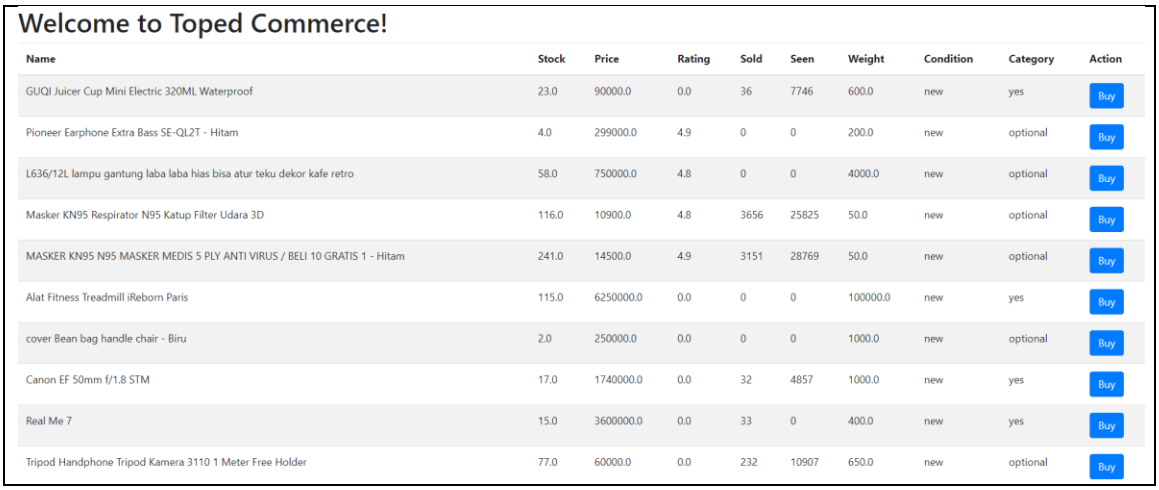


Welcome to Toped Commerce!

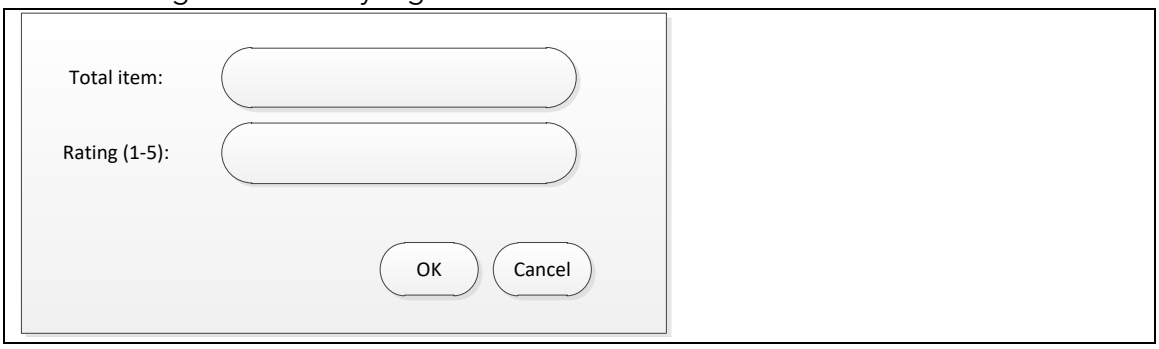
Name	Stock	Price	Rating	Sold	Seen	Weight	Condition	Category
GUQI Juicer Cup Mini Electric 320ML Waterproof	23.0	90000.0	0.0	36	7746	600.0	new	yes
Pioneer Earphone Extra Bass SE-QL2T - Hitam	4.0	299000.0	4.9	0	0	200.0	new	optional
L636/12L lampu gantung laba laba hias bisa atur teku dekor kafe retro	58.0	750000.0	4.8	0	0	4000.0	new	optional
Masker KN95 Respirator N95 Katup Filter Udara 3D	116.0	10900.0	4.8	3656	25825	50.0	new	optional
MASKER KN95 N95 MASKER MEDIS 5 PLY ANTI VIRUS / BELI 10 GRATIS 1 - Hitam	241.0	14500.0	4.9	3151	28769	50.0	new	optional
Alat Fitness Treadmill iReborn Paris	115.0	6250000.0	0.0	0	0	100000.0	new	yes
cover Bean bag handle chair - Biru	2.0	250000.0	0.0	0	0	1000.0	new	optional
Canon EF 50mm f/1.8 STM	17.0	1740000.0	0.0	32	4857	1000.0	new	yes
Real Me 7	15.0	3600000.0	0.0	33	0	400.0	new	yes
Tripod Handphone Tripod Kamera 3110 1 Meter Free Holder	77.0	60000.0	0.0	232	10907	650.0	new	optional

Tugas

1. Masih melanjutkan toped web app di atas, buatlah mekanisme pembelian dengan menambahkan sebuah link beli untuk masing-masing item, seperti pada gambar di bawah:



Ketika tombol “Buy” diklik, maka akan muncul pop up seperti di bawah ini, dan user diminta mengisi data-data yang diminta.



Secara otomatis jumlah mula-mula stock akan terupdate sesuai dengan **total item** yang dibeli, termasuk juga **rating**, sold, dan seen. Perhitungan rating adalah rating mula-mula ditambah rating inputan user dibagi 2.

2. Kembangkanlah toped web app diatas sehingga mengizinkan adanya user management. Hanya user biasa (guest) yang dapat melakukan pembelian, sedangkan user admin dapat menambah/mengupdate stock, weight, condition dan category. Beri tombol **Update** pada kolom **Action** apabila user yang login adalah admin. Untuk mempermudah pengembangan aplikasi ini, updating data oleh admin hanya lewat pop up.

3. Sesuai dengan latihan di atas sebelumnya dimana anda telah membuat collection store, tugas anda adalah menghubungkan antara collection store dengan item. Hint: mirip dengan item dengan itemdetail. Lalu munculkan ke dalam web toped anda.