

MidEng GK8.1 Spring Data and ORM

Stefan Kordov 4DHIT

First i forked this github-repo:

https://github.com/ThomasMicheler/DEZSYS_GK862_DATAWAREHOUSE_ORM.git

Then i copied the Classes from the past Warehouse-Project:

```
package org.example;

import jakarta.persistence.*;
import java.time.LocalDateTime;
import java.util.List;
public class WarehouseStock {
    private Integer warehouseId;
    public String warehouseName;
    public String timestamp;
    public String warehouseCountry;
    public String warehouseCity;
    public String address;
    public List<ProductData> productData;

    public WarehouseStock() {}

    public WarehouseStock(Integer warehouseId,
                          String warehouseName,
                          String timestamp,
                          String warehouseCountry,
                          String warehouseCity,
                          String address,
                          List<ProductData> productData) {

        this.warehouseId = warehouseId;
```

```

        this.warehouseName = warehouseName;
        this.timestamp = timestamp;
        this.warehouseCountry = warehouseCountry;
        this.warehouseCity = warehouseCity;
        this.address = address;
        this.productData = productData;
    }
}

```

```

package org.example;

import jakarta.persistence.*;
public class ProductData {
    public String productId;
    public String productName;
    public String productCategory;
    public int productAmount;
    public String productUnit;

    public ProductData() {}

    public ProductData(String productId,
                       String productName,
                       String productCategory,
                       int productAmount,
                       String productUnit) {
        this.productId = productId;
        this.productName = productName;
        this.productCategory = productCategory;
        this.productAmount = productAmount;
        this.productUnit = productUnit;
    }
}

```

Then i implemented these into Spring:

First i added the `@Entity` :

```
@Entity  
public class WarehouseStock {
```

Then I labeled the ID :

```
@Id  
@GeneratedValue(strategy= GenerationType.AUTO)  
private Integer warehouseId;
```

And the Collection as an ElementCollection:

```
@ElementCollection  
public List<ProductData> productData;
```

For ProductData i added almost the same things:

```
@Embeddable  
public class ProductData {  
    @GeneratedValue(strategy= GenerationType.AUTO)  
    public String productId;
```

Then i created a WarehouseStockRepository Interface :

```
package org.example;  
  
import org.springframework.data.repository.CrudRepository;  
  
public interface WarehouseStockRepository  
    extends CrudRepository<WarehouseStock, String> {  
  
}
```

And lastly i created a method for creating the warehouse:

```
@PostMapping(path="/addWh")  
public @ResponseBody String addNewWh(  
    @RequestParam String productname,  
    @RequestParam String whname,
```

```

    @RequestParam int warehouseAnzahl,
    @RequestParam int productAnzahl) {

    for (int w = 1; w <= warehouseAnzahl; w++) {

        List<ProductData> products = new LinkedList<>();

        for (int p = 1; p <= productAnzahl; p++) {
            ProductData product = new ProductData(
                String.valueOf(p),
                productname + p,
                "String productCategory",
                1,
                "String productUnit"
            );
            products.add(product);
        }

        WarehouseStock warehouse = new WarehouseStock(
            w,
            whname + w,
            "String timestamp",
            "String warehouseCountry",
            "String warehouseCity",
            "String address",
            products
        );

        warehouseRepository.save(warehouse);
    }

    return "Saved";
}

```

Now i try it with the following POST-Request:

```
[stefan@MacBook-Pro-von-Stefan ~ % curl -X POST "http://localhost:8080/demo/addWh"
" -d "productname=stefans produkt" -d "whname=stefans warenhaus" -d "warehouseAnzahl=2" -d "productAnzahl=5"
Saved%
stefan@MacBook-Pro-von-Stefan ~ % ]
```

Then i check the tables and yep, looks good:

WHERE					ORDER BY	
	address	timestamp	warehouse_city	warehouse_country	warehouse_name	
1	String address	String timestamp	String warehouseCity	String warehouseCountry	stefans warenhaus1	
2	String address	String timestamp	String warehouseCity	String warehouseCountry	stefans warenhaus2	

WHERE					ORDER BY	
	product_amount	product_category	product_id	product_name	product_unit	
1	1	String productCategory	1	stefans produkt1	String productUnit	
2	1	String productCategory	2	stefans produkt2	String productUnit	
3	1	String productCategory	3	stefans produkt3	String productUnit	
4	1	String productCategory	4	stefans produkt4	String productUnit	
5	1	String productCategory	5	stefans produkt5	String productUnit	
6	1	String productCategory	1	stefans produkt1	String productUnit	
7	1	String productCategory	2	stefans produkt2	String productUnit	
8	1	String productCategory	3	stefans produkt3	String productUnit	
9	1	String productCategory	4	stefans produkt4	String productUnit	
10	1	String productCategory	5	stefans produkt5	String productUnit	

Perfect!

Collect all data of one data warehouse specified by datawarehouseID

First i went back to my WarehouseStockRepository interface and i added this

```
WarehouseStock findByWarehouseId(Integer warehouseID);
```

And in the MainController Class a simple:

```
@GetMapping("/warehouse/{id}")
    public WarehouseStock getWarehouseData(@PathVariable int
id) {
    return warehouseRepository.findByWarehouseId(id);
}
```

Collect a single product of a data warehouse specified by datawarehouseID and productID

For this Task i went into MainController and wrote this:

```
@GetMapping("/{warehouseId}/{productId}")
public @ResponseBody ProductData getSingleProduct(
    @PathVariable Integer warehouseId,
    @PathVariable String productId) {

    WarehouseStock warehouse =
        warehouseRepository.findByWarehouseId(wareho
useId);

    if (warehouse == null) {
        return null;
    }

    for (ProductData product : warehouse.productData) {
        if (product.productId.equals(productId)) {
            return product;
        }
    }
}
```

```
        return null;  
    }  
}
```

Update a data warehouse using datawarehouseID

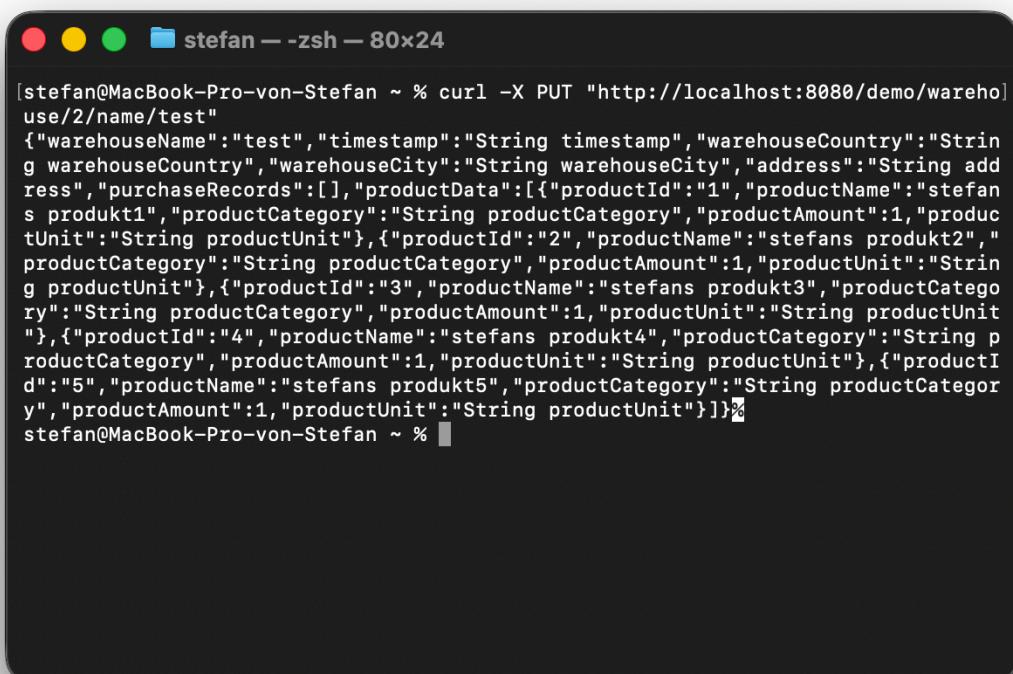
For this Task i wrote this Method:

```
@PutMapping("/warehouse/{warehouseId}/name/{newName}")  
public @ResponseBody WarehouseStock updateWarehouseName(  
    @PathVariable Integer warehouseId,  
    @PathVariable String newName) {  
  
    WarehouseStock existing =  
        warehouseRepository.findById(wareho  
useId);  
  
    if (existing == null) {  
        return null;  
    }  
  
    existing.warehouseName = newName;  
  
    return warehouseRepository.save(existing);  
}
```

Now i can test these three:

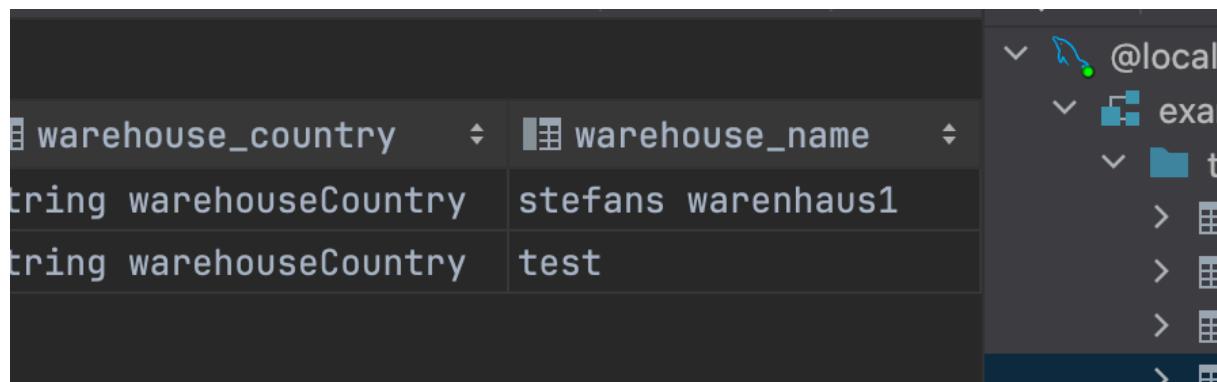
```
[stefan@MacBook-Pro-von-Stefan ~ % curl -X GET "http://localhost:8080/demo/warehouse/53"
{"warehouseName":"stefanswarenhasu","timestamp":"String timestamp","warehouseCountry":"String warehouseCountry","warehouseCity":"String warehouseCity","address":"String address","purchaseRecords":[],"productData":[{"productId":1,"productName":"stefans produkt1","productCategory":"String productCategory","productAmount":1,"productUnit":"String productUnit"}, {"productId":2,"productName":"stefans produkt2","productCategory":"String productCategory","productAmount":1,"productUnit":"String productUnit"}, {"productId":3,"productName":"stefans produkt3","productCategory":"String productCategory","productAmount":1,"productUnit":"String productUnit"}, {"productId":4,"productName":"stefans produkt4","productCategory":"String productCategory","productAmount":1,"productUnit":"String productUnit"}, {"productId":5,"productName":"stefans produkt5","productCategory":"String productCategory","productAmount":1,"productUnit":"String productUnit"}]}%]
stefan@MacBook-Pro-von-Stefan ~ %
```

```
[stefan@MacBook-Pro-von-Stefan ~ % curl -X GET "http://localhost:8080/demo/2/1" ]
[{"productId":1,"productName":"stefans produkt1","productCategory":"String productCategory","productAmount":1,"productUnit":"String productUnit"}]%
stefan@MacBook-Pro-von-Stefan ~ %
```



```
[stefan@MacBook-Pro-von-Stefan ~ % curl -X PUT "http://localhost:8080/demo/warehouse/2/name/test"
{"warehouseName":"test","timestamp":"String timestamp","warehouseCountry":"String warehouseCountry","warehouseCity":"String warehouseCity","address":"String address","purchaseRecords":[],"productData":[{"productId": "1", "productName": "stefans produkt1", "productCategory": "String productCategory", "productAmount": 1, "productUnit": "String productUnit"}, {"productId": "2", "productName": "stefans produkt2", "productCategory": "String productCategory", "productAmount": 1, "productUnit": "String productUnit"}, {"productId": "3", "productName": "stefans produkt3", "productCategory": "String productCategory", "productAmount": 1, "productUnit": "String productUnit"}, {"productId": "4", "productName": "stefans produkt4", "productCategory": "String productCategory", "productAmount": 1, "productUnit": "String productUnit"}, {"productId": "5", "productName": "stefans produkt5", "productCategory": "String productCategory", "productAmount": 1, "productUnit": "String productUnit"}]}%
```

stefan@MacBook-Pro-von-Stefan ~ %



	warehouse_name	warehouse_country
	stefans warenhaus1	string warehouseCountry
	test	string warehouseCountry

Perfect! Everything works!

Extend the data model with data about the customer purchases

First i made a new Class called `Einkauf`

```
package org.example;

import jakarta.persistence.Embeddable;
import java.util.Date;
```

```

@Embeddable
public class Einkauf {

    public String productId;

    public String purchaseDate;

    public Einkauf(String productId) {
        this.productId = productId;
        this.purchaseDate = "2026-02-24";
    }

    public Einkauf() {
        this.purchaseDate = "2026-02-24";
    }
}

```

Then in warehouseStock i added a:

```

@ElementCollection
public List<Einkauf> purchaseRecords;

```

At last i wrote this script:

```

@PostMapping("/warehouse/{warehouseId}/purchase/{item}")
public @ResponseBody WarehouseStock addPurchase(
    @PathVariable Integer warehouseId,
    @PathVariable String item) {

    Einkauf newPurchase = new Einkauf(item);
    WarehouseStock warehouse =
        warehouseRepository.findById(wareho
useId);

    if (warehouse == null) {
        return null;
    }
    if (warehouse.purchaseRecords == null) {

```

```

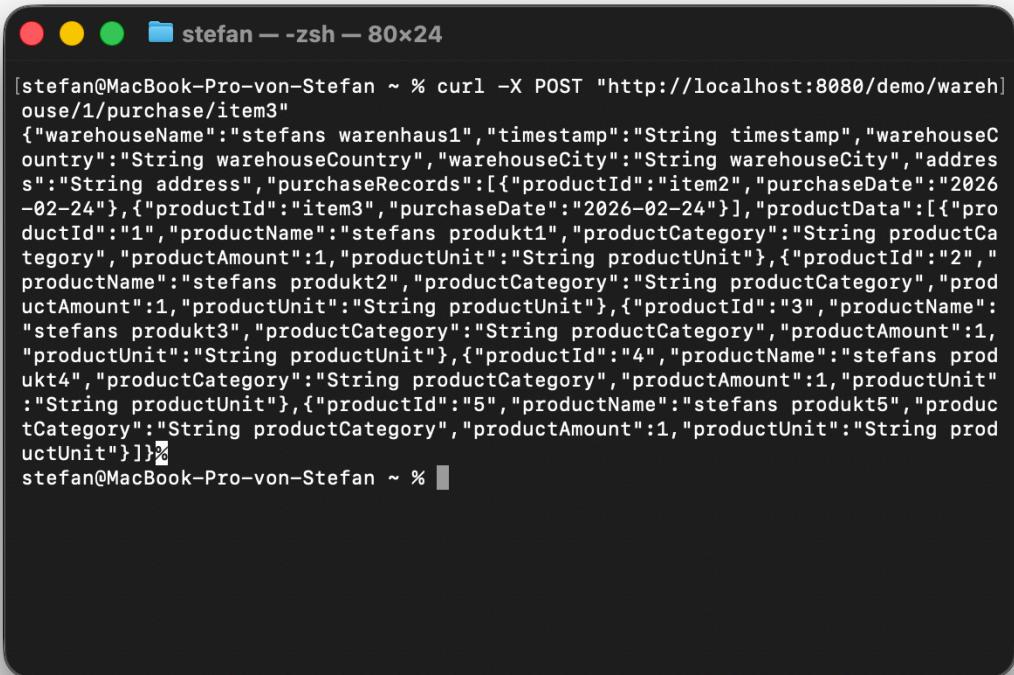
        warehouse.purchaseRecords = new ArrayList<>();
    }

    warehouse.purchaseRecords.add(newPurchase);

    return warehouseRepository.save(warehouse);
}

```

Now with a simple:



```

[stefan@MacBook-Pro-von-Stefan ~ % curl -X POST "http://localhost:8080/demo/warehouse/1/purchase/item3"
{"warehouseName": "stefans warenhaus1", "timestamp": "String timestamp", "warehouseCountry": "String warehouseCountry", "warehouseCity": "String warehouseCity", "address": "String address", "purchaseRecords": [{"productId": "item2", "purchaseDate": "2026-02-24"}, {"productId": "item3", "purchaseDate": "2026-02-24"}], "productData": [{"productId": "1", "productName": "stefans produkt1", "productCategory": "String productCategory", "productAmount": 1, "productUnit": "String productUnit"}, {"productId": "2", "productName": "stefans produkt2", "productCategory": "String productCategory", "productAmount": 1, "productUnit": "String productUnit"}, {"productId": "3", "productName": "stefans produkt3", "productCategory": "String productCategory", "productAmount": 1, "productUnit": "String productUnit"}, {"productId": "4", "productName": "stefans produkt4", "productCategory": "String productCategory", "productAmount": 1, "productUnit": "String productUnit"}, {"productId": "5", "productName": "stefans produkt5", "productCategory": "String productCategory", "productAmount": 1, "productUnit": "String productUnit"}]}
stefan@MacBook-Pro-von-Stefan ~ %

```

I see this:

WHERE	ORDER BY warehouse_stock_warehouse_id		
warehouse_stock_warehouse_id	product_id	purchase_date	
1	item2	2026-02-24	
1	item3	2026-02-24	

