

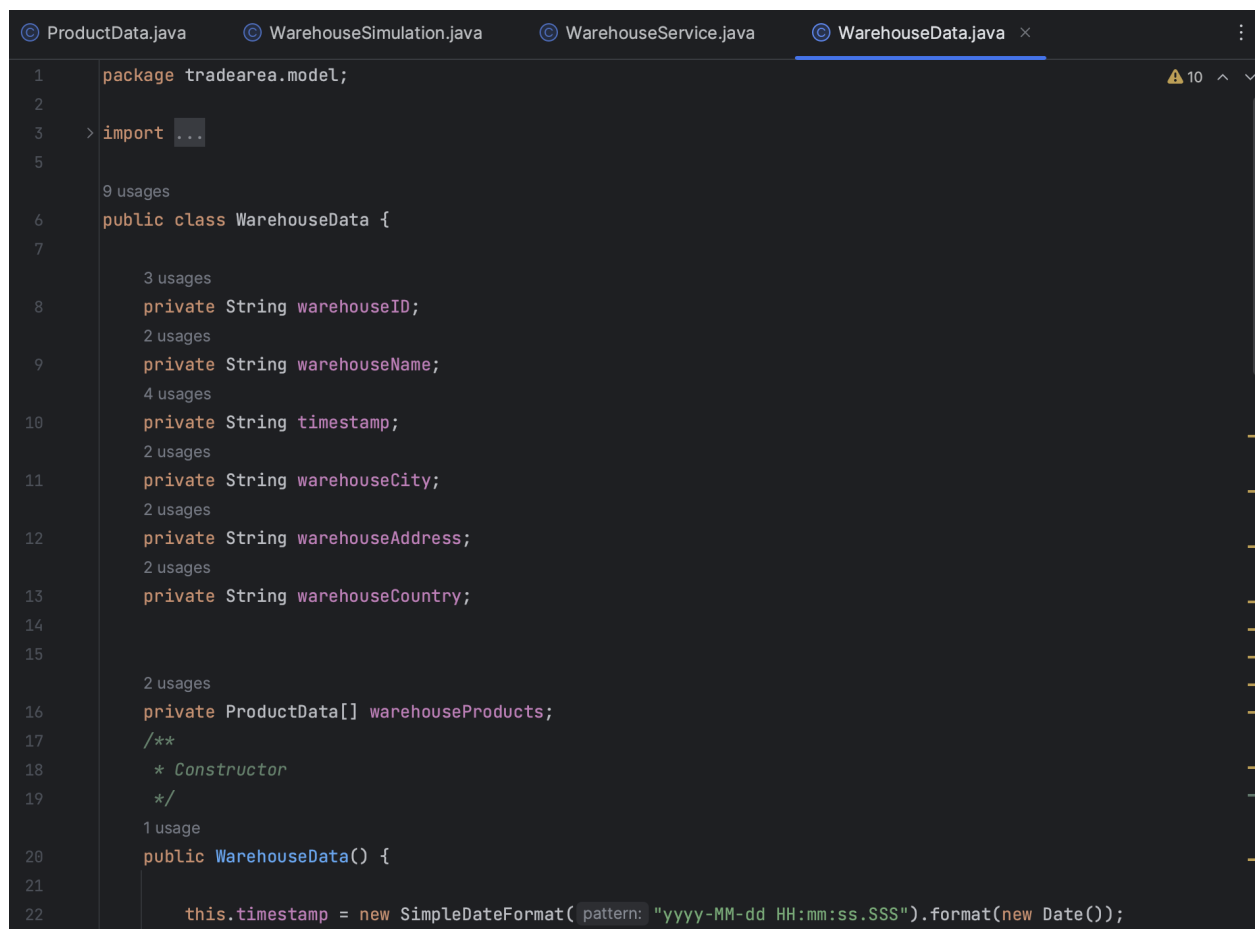
MidEng 7.1 Protokoll

@Matteo Jozepovic

@September 19, 2023

Bis GKÜ: Warehouse REST & Dataformats

Zuerst erweitere ich die Daten in WarehouseData mit z.b. City, Address und Country.



```
1 package tradearea.model;
2
3 > import ...
4
5 9 usages
6 public class WarehouseData {
7
8     3 usages
9     private String warehouseID;
10    2 usages
11    private String warehouseName;
12    4 usages
13    private String timestamp;
14    2 usages
15    private String warehouseCity;
16    2 usages
17    private String warehouseAddress;
18    2 usages
19    private String warehouseCountry;
20
21    2 usages
22    private ProductData[] warehouseProducts;
23    /**
24     * Constructor
25     */
26    1 usage
27    public WarehouseData() {
28
29        this.timestamp = new SimpleDateFormat(pattern: "yyyy-MM-dd HH:mm:ss.SSS").format(new Date());
30    }
31 }
```

Außerdem erstelle ich den Array `warehouseProducts` um die einzelnen Produkte im Warenhaus ebenfalls anzeigen zu lassen. Die Unterdaten für die Produkte werden dann separat in einer neuen Klasse namens ProductData geschrieben. Für alle Attribute erstelle ich außerdem Getter und Setter.

```
ProductData.java x WarehouseSimulation.java WarehouseService.java WarehouseData.java
1 package tradearea.model;
2
3 public class ProductData {
4     2 usages
5     private String productId;
6     2 usages
7     private String productName;
8     2 usages
9     private String productQuantity;
10    2 usages
11    private String productUnit;
12
13    no usages
14    public String getProductId() {
15        return productId;
16    }
17
18    1 usage
19    public void setProductId(String productId) {
20        this.productId = productId;
21    }
22
23    1 usage
24    public String getProductName() {
25        return productName;
26    }
27
28    1 usage
```

Die Methode `generateRandomWarehouseData()` generiert mir eine Random ID, das passende Datum(sformat) und setzt fix den Namen und Land des Warehouses.

```

11 public class WarehouseSimulation {
12
13     1 usage
14     public static WarehouseData generateRandomWarehouseData() {
15         WarehouseData warehouseData = new WarehouseData();
16
17         warehouseData.setWarehouseName("NOE Korneuburg");
18         warehouseData.setWarehouseCountry("Austria");
19
20         Random random = new Random();
21         warehouseData.setWarehouseID(String.format("%03d", random.nextInt( bound: 1000)));
22
23         SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss.SSS");
24         warehouseData.setTimestamp(dateFormat.format(new Date()));
25
26         ProductData[] productData = new ProductData[2];
27         productData[0] = generateRandomProductData();
28         productData[1] = generateRandomProductData();
29
30         warehouseData.setWarehouseProducts(productData);
31
32         return warehouseData;
33     }

```

Die `generateRandomProductData()` erstellt mir zufällige Daten für ID, Name, Kategorie, Stückzahl und Einheit.

```

38 @
39 public static ProductData generateRandomProductData() {
40     ProductData productData = new ProductData();
41     Random random = new Random();
42
43     productData.setProductId(String.format("PD%03d", random.nextInt( bound: 1000)));
44     productData.setProductName(generateRandomProductName());
45     productData.setProductCategory(assignProductCategory(productData.getProductName()));
46     productData.setProductQuantity(Integer.toString(random.nextInt( bound: 1000)));
47     productData.setProductUnit("pcs");
48
49     return productData;
50 }

```

`generateRandomProductName()` Gibt mir ein zufälliges Produkt zurück und `assignProductCategory()` weist dem jeweiligen Produkt die richtige Kategorie zu.

```
ProductData.java WarehouseConsumer.java WarehouseSimulation.java x WarehouseService.java Warel v
49 }
50
51 1 usage
52 public static String generateRandomProductName() {
53     // You can replace this with a list of actual product names
54     String[] productNames = {"Coca-Cola", "Pepsi", "Fanta", "Almdudler", "Voeslauer", "Coffee"};
55     Random random = new Random();
56     return productNames[random.nextInt(productNames.length)];
57 }
58
59 1 usage
60 @ public static String assignProductCategory(String productName) {
61     if (productName.equals("Coca-Cola") || productName.equals("Pepsi")) {
62         return "Soda";
63     } else if (productName.equals("Fanta") || productName.equals("Almdudler")) {
64         return "Juice";
65     } else if (productName.equals("Voeslauer")) {
66         return "Water";
67     } else if (productName.equals("Coffee")) {
68         return "Coffee";
69     } else {
70         return "Unknown";
71     }
72 }
73 }
```

Zum Schluss können wir uns jetzt die Daten auf localhost abrufen. Uns steht die Darstellung in JSON oder XML zur Verfügung, bei XML (aufgrund eines anschaulicheren hierarchie Systems) kann man die Daten einfacher lesen.

← → ↻ 🏠 ⓘ localhost:8080/warehouse/001/xml

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼ <WarehouseData>
  <warehouseID>764</warehouseID>
  <warehouseName>Wien Jedlersdorf</warehouseName>
  <timestamp>2023-09-20 12:39:39.915</timestamp>
  <warehouseCity/>
  <warehouseAddress/>
  <warehouseCountry>Austria</warehouseCountry>
  ▼ <warehouseProducts>
    ▼ <warehouseProducts>
      <productId>PD506</productId>
      <productName>Coffee</productName>
      <productQuantity>139</productQuantity>
      <productUnit>pcs</productUnit>
      <productCategory>Coffee</productCategory>
    </warehouseProducts>
    ▼ <warehouseProducts>
      <productId>PD871</productId>
      <productName>Voeslauer</productName>
      <productQuantity>796</productQuantity>
      <productUnit>pcs</productUnit>
      <productCategory>Water</productCategory>
    </warehouseProducts>
  </warehouseProducts>
</WarehouseData>
```