MidEng_7.1

Verfasser: **Leonhard Stransky, 4AHIT**

Datum: **07.11.2023**

Einführung

Der Lagerstandort Linz der grossen Handelskette KONSUM lagert alle Lebensmittel, Haushaltswaren und Hygieneartikel fuer die Filialen in Wien, Niederoesterreich und Burgenland. Das Lagerprogramm vor Ort soll um eine Schnittstelle erweitert werden, damit die Konzernzentrale alle Lagerdaten tagesaktuell abrufen kann. Um die Moeglichkeiten der Anbindung moeglichsts flexibel zu gestalten, sollen die Datenformate JSON und XML unterstuetzt werden.

Aufgabenstellung:

Entwickeln Sie einen Simulator, der die Daten eines Lagerstandortes (WHx) generiert. Es ist dabei zu achten, dass die Daten realistisch sind und im Zusammenhang mit entsprechenden Einheiten erzeugt werden.

Diese Daten sollen gemeinsam mit einigen Details zum dem Standort ueber eine REST Schnittstelle veroeffentlicht werden. Die Schnittstelle verwendet standardmaessig das JSON Format und kann optional auf XML umgestellt werden.

Praxis:

Gradle File:

(build.gradle)

```
plugins {
    id 'org.springframework.boot' version '2.5.4'
    id 'io.spring.dependency-management' version '1.0.11.RELEASE'
    id 'java'
}

group = 'dezsys'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '11'

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-rest'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'com.fasterxml.jackson.dataformat:jackson-dataformat-xml'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}
```

```
test {
    useJUnitPlatform()
}
```

Dependencies:

org.springframework.boot:spring-boot-starter-data-rest: Ermöglicht die einfache Erstellung von RESTful Webdiensten mit Spring Boot.

org.springframework.boot:spring-boot-starter-web: Bietet Unterstützung für die Entwicklung von Webanwendungen mit Spring Boot.

com.fasterxml.jackson.dataformat:jackson-dataformat-xml: Erlaubt die Verarbeitung von XML-Dateien in der Anwendung mit Jackson Dataformat.

org.springframework.boot:spring-boot-starter-test: Stellt Tools und Bibliotheken für das Testen von Spring Boot-Anwendungen zur Verfügung.

WarehouseController

Diese Klasse ist die Hauptklasse des Projekts. Sie beinhaltet die Main-Methode und die Methoden, die die REST Schnittstelle implementieren.

```
@RequestMapping("/")
    public String warehouseMain() {
        String mainPage = "This is the warehouse application!
(DEZSYS_WAREHOUSE_REST) <br/><br/>" +
                          "<a href='http://localhost:8080/warehouse/001/data'>Link
to warehouse/001/data</a><br/>" +
                          "<a href='http://localhost:8080/warehouse/001/xml'>Link
to warehouse/001/xml</a><br/>" +
                          "<a
href='http://localhost:8080/warehouse/001/transfer'>Link to
warehouse/001/transfer</a><br/>";
        return mainPage;
    }
```

Diese Methode beinhaltet die HTML Seite, die beim Aufruf von localhost:8080 angezeigt wird.

```
@RequestMapping(value="/warehouse/{inID}/data", produces =
MediaType.APPLICATION_JSON_VALUE)
    public WarehouseData warehouseData( @PathVariable String inID ) {
        return service.getWarehouseData( inID );
    }

    @RequestMapping(value="/warehouse/{inID}/xml", produces =
MediaType.APPLICATION_XML_VALUE)
    public WarehouseData warehouseDataXML( @PathVariable String inID ) {
```

```
        return service.getWarehouseData( inID );
    }

    @RequestMapping("/warehouse/{inID}/transfer")
    public String warehouseTransfer( @PathVariable String inID ) {
        return service.getGreetings("Warehouse.Transfer!");
    }
```

Diese Methoden sind REST-Endpunkte die über einen GET-Request erreichbar sind. Sie geben die Daten des Lagers zurück. In Form von JSON oder XML. Der Transfer Endpunkt gibt nur einen String zurück.

WarehouseSimulation

```
public WarehouseData getData( String inID ) {

        WarehouseData data = new WarehouseData();
        data.setWarehouseID( inID );
        data.setWarehouseName( "Linz Bahnhof" );

        return data;

    }
```

In der WarehouseSimulation Klasse werden die Daten des Lagers initalisiert und diese Methode wird verwendet um die Daten zurückzugeben.

WarehouseService

```
@Service
public class WarehouseService {

    public String getGreetings( String inModule ) {
        return "Greetings from " + inModule;
    }

    public WarehouseData getWarehouseData( String inID ) {

        WarehouseSimulation simulation = new WarehouseSimulation();
        return simulation.getData( inID );

    }

}
```

Hier werden die Methoden der WarehouseSimulation Klasse aufgerufen und die Daten zurückgegeben.

Sonstiges:

Ich habe noch zwei Klassen erstellt. Eine zur generierung von den Produkt Daten erstellt und das erstellen von einer ArrayList mit den Produkten. Und eine Klasse die die Daten des Lagers beinhaltet.

Json File:


```
{"warehouseID":"001","warehouseName":"Linz
Bahnhof","warehouseAddress":"Bahnhofsstrasse
27/9","warehousePostalCode":"Linz","warehouseCity":"Linz","warehouseCountry":"Aust
ria","timestamp":"2023-11-07 15:33:51.088","productData":{"products":
[{"productID":"136-
26818415","productName":"Cheese","productCategory":"Food","productQuantity":1131,"
productUnit":"1 KG"},{"productID":"468-
35127549","productName":"Beer","productCategory":"Drink","productQuantity":3372,"p
roductUnit":"1 L"},{"productID":"562-
32950783","productName":"Cheese","productCategory":"Drink","productQuantity":3980,
"productUnit":"1 L"},{"productID":"136-
94370052","productName":"Burger","productCategory":"Food","productQuantity":301,"p
roductUnit":"1 KG"}]}}
```


XML File:


```
<WarehouseData>
<warehouseID>001</warehouseID>
<warehouseName>Linz Bahnhof</warehouseName>
<warehouseAddress>Bahnhofsstrasse 27/9</warehouseAddress>
<warehousePostalCode>Linz</warehousePostalCode>
<warehouseCity>Linz</warehouseCity>
<warehouseCountry>Austria</warehouseCountry>
<timestamp>2023-11-07 15:34:20.299</timestamp>
<productData>
<products>
<products>
<productID>890-10477519</productID>
<productName>Beer</productName>
<productCategory>Drink</productCategory>
<productQuantity>198</productQuantity>
<productUnit>1 L</productUnit>
</products>
<products>
<productID>680-30712564</productID>
<productName>Coconut</productName>
<productCategory>Drink</productCategory>
<productQuantity>536</productQuantity>
<productUnit>1 L</productUnit>
</products>
<products>
<productID>533-65634859</productID>
<productName>Limomade</productName>
<productCategory>Food</productCategory>
<productQuantity>1457</productQuantity>
```

```
<productUnit>1 KG</productUnit>
</products>
<products>
<productID>899-81494741</productID>
<productName>Limomade</productName>
<productCategory>Drink</productCategory>
<productQuantity>2577</productQuantity>
<productUnit>1 L</productUnit>
</products>
</products>
</productData>
</WarehouseData>
```

Quellen:

[1] www.Github.com 2023. [online] Available at: https://github.com/ThomasMicheler/DEZSYS_GK771_WAREHOUSE_REST.git [Accessed 07 November 2023].

[2] elearning.tgm.ac.at 2023. [online] Available at: https://elearning.tgm.ac.at/mod/resource/view.php?id=79890 [Accessed 18 March 2023].