MidEng 7.1 Warehouse REST & Dataformats

author: Felix Aust 4AHIT
date: 19.9.2023

JSON & XML

```
@RestController
public class WarehouseController {
```

Using the @RestController tag i declare this class as the Controller for any requests.

```
    @RequestMapping("/")
    public String warehouseMain() {
        String mainPage = "This is the warehouse application!
(DEZSYS_WAREHOUSE_REST) <br/><br/>" +
                "<a
href='http://localhost:8080/api/warehouse/json'>JSON</a><br/>" +
                "<a href='http://localhost:8080/api/warehouse/xml'>XML</a>
<br/>" +
                "<a href='http://localhost:8081/'>Consumer</a><br/>";
        return mainPage;
    }
```

With the @RequestMapping and the path "/" I create a main Page with links to all other pages.

```
    @GetMapping(value = "/api/warehouse/json", produces =
"application/json")
    public String getWarehouseDataJSON() {
        WarehouseData warehouseData = new WarehouseData();
        ObjectMapper objectMapper = new ObjectMapper();
        String json = "error";
        try {
            json = objectMapper.writeValueAsString(warehouseData);
            System.out.println(json);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return json;
    }

    @RequestMapping(value="/api/warehouse/xml", produces =
MediaType.APPLICATION_XML_VALUE)
    public WarehouseData getWarehouseDataXML() {
        WarehouseData warehouseData = new WarehouseData();
        return warehouseData;
```

```
    }
}
```

I used 2 variants to convert the java object.

1. Using a objectMapper to convert it to json
2. Using the produces = MediaType which automatically converts it to the specified format, in this case xml

```
public class WarehouseData {
    private String warehouseID;
    private String warehouseName;
    private String warehouseAddress;
    private String warehousePostalCode ;
    private String warehouseCity;
    private String warehouseCountry;
    private String timestamp;
    private ProductData productData;

    public WarehouseData() {
        warehouseID = "001";
        warehouseName = "Linz Bahnhof";
        warehouseAddress = "Bahnhofsstrasse 27/9";
        warehousePostalCode = "Linz";
        warehouseCity = "Linz";
        warehouseCountry = "Austria";
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSS");
        this.timestamp = dateFormat.format(new Date());
        this.productData = new ProductData(4);
    }
    .
    .
    .
```

The WarehouseData class has fixed warehouse information and randomly generated products using the ProductData class with an integer parameter for the amount of products wanted. In addition it also needs getters for the ObjectMapper to work.

```
public class ProductData{
    private List<Product> products;


    public ProductData(int amount) {
        this.products = new ArrayList<>();
        for (int i = 0; i < amount; i++) {
            this.products.add(new Product());
        }
    }
```

```
    public List<Product> getProducts() {
        return products;
    }
}
```

ProductData uses a simple for loop for the products.

```
private static final String[] UNIQUE_NAMES = {"Pineapple", "Quinoa",
"Tofu", "Mussels", "Gouda cheese", "Cabbage", "Sushi", "Pistachios",
"Guacamole", "Croissant"};
    private static final String[] UNIQUE_CATEGORIES = {"Drink", "Food"};

    public Product() {
        Random random = new Random();
        this.productID = String.valueOf(random.nextInt(899)+100)+"-
"+String.valueOf(random.nextInt(89999999)+10000000);
        this.productName =
UNIQUE_NAMES[random.nextInt(UNIQUE_NAMES.length)];
        this.productCategory =
UNIQUE_CATEGORIES[random.nextInt(UNIQUE_CATEGORIES.length)];
        this.productQuantity = random.nextInt(4001 - 100) + 100;
        if(productCategory.equals("Drink")){
            this.productUnit = "1 L";
        } else {
            this.productUnit = "1 KG";
        }

    }
```

In the Product class i use a fixed list of names and categories that get picked randomly. Depending on the category the unit either is L or KG.

```
@SpringBootApplication
public class WarehouseKonsumApplication {

    public static void main(String[] args) {
        SpringApplication.run(WarehouseKonsumApplication.class, args);
    }

    @Configuration
    public class CorsConfig {
        @Bean
        public CorsFilter corsFilter() {
            CorsConfiguration corsConfig = new CorsConfiguration();
            corsConfig.addAllowedOrigin("*");

            UrlBasedCorsConfigurationSource source = new
UrlBasedCorsConfigurationSource();
            source.registerCorsConfiguration("/**", corsConfig);
```

```
            return new CorsFilter(source);
        }
    }
}
```

Simple springbootapplication start but with the addition of Cors seetings to allow the consumer to request the data.

Consumer

```
document.addEventListener("DOMContentLoaded", function () {

    fetch("http://localhost:8080/api/warehouse/json")
        .then(response => response.json())
        .then(data => {

            const products = data.productData.products;

            const table = document.getElementById("warehouse-table");

            const headers = Object.keys(products[0]);
            const headerRow = table.insertRow();
            headers.forEach(headerText => {
                const header = document.createElement("th");
                header.textContent = headerText;
                headerRow.appendChild(header);
            });

            products.forEach(product => {
                const row = table.insertRow();
                headers.forEach(headerText => {
                    const cell = row.insertCell();
                    cell.textContent = product[headerText];
                });
            });
        })
        .catch(error => {
            console.error(error);
        });
});
```

Fetch makes a GET request to the URL of the warehouse with the json data. Afterwards I get the products from the json and create a table with the amount of rows needed for the products. I create a header row using "th" and add all products using a forEach loop. All data gets added to a table in the html by creating a reference with "document.getElementById".

In the html i only have to link the script and execute the html. In my case i executed "python3 -m http.server 8081" in the directory of my index.html to host it on port 8081.