Middleware Engineering "REST and Data Formats"

Aufgabenstellung

Siehe TASK.md

Implementierung

Architektur

Die Anwendung folgt der Schichtenarchitektur:

1. **Model Layer** (`rest.model`)
2. **Service Layer** (`rest.warehouse.WarehouseService`)
3. **Controller Layer** (`rest.warehouse.WarehouseController`)
4. **Simulation Layer** (`rest.warehouse.WarehouseSimulation`)

Wichtigste Arbeitsschritte

2. Datenmodell-Implementierung

WarehouseData.java:

- Hauptmodell für Lagerdaten
- Verwendet Jackson XML-Annotationen für XML-Serialisierung:
  - `@JacksonXmlRootElement(localName = "warehouseData")` - definiert Root-Element
  - `@JacksonXmlElementWrapper(useWrapping = false)` - verhindert zusätzliche Wrapper-Elemente
  - `@JacksonXmlProperty(localName = "ProductData")` - benennt XML-Elemente

ProductData.java:

- Modell für einzelne Produkte
- Enthält: ID, Name, Kategorie, Menge, Einheit

ProductTemplate.java:

- Template-Klasse für realistische Produktgenerierung

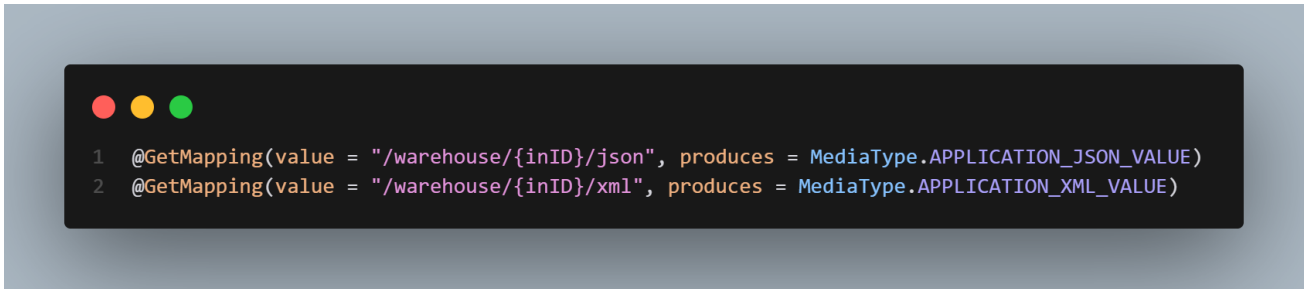
- Definiert Produkttypen mit realistischen Mengenbereichen und Einheiten

3. REST Controller

WarehouseController.java

Wichtige Erkenntnisse zu REST Controllern:

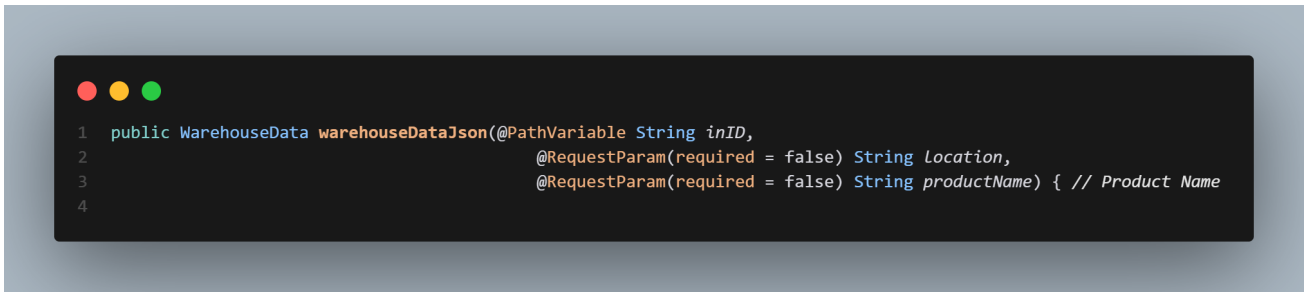
1. **@RestController Annotation:**
   - Kombiniert `@Controller` und `@ResponseBody`
   - Alle Methoden-Rückgabewerte werden automatisch serialisiert (JSON/XML)
   - Keine Notwendigkeit für explizite `@ResponseBody` Annotationen
2. **@CrossOrigin(origins = "*"):**
   - Ermöglicht CORS (Cross-Origin Resource Sharing)
   - Wichtig für Frontend-Anwendungen, die von anderen Domains aufrufen
   - `origins = "*"` erlaubt alle Origins
3. **MediaType-basierte Endpunkte***:



```
@GetMapping(value = "/warehouse/{inID}/json", produces = MediaType.APPLICATION_JSON_VALUE)
@GetMapping(value = "/warehouse/{inID}/xml", produces = MediaType.APPLICATION_XML_VALUE)
```

- Zwei separate Endpunkte für JSON und XML
- `produces` definiert den Content-Type der Antwort
- Spring Boot wählt automatisch den passenden MessageConverter aus

4. **Path Variables und Request Parameters:**



```
public WarehouseData warehouseDataJson(@PathVariable String inID,
                                       @RequestParam(required = false) String location,
                                       @RequestParam(required = false) String productName) { // Product Name

```

- `@PathVariable`: Extrahiert Werte aus URL-Pfad (`/warehouse/{inID}/json`)
- `@RequestParam`: Extrahiert Query-Parameter (`?location=Linz&productName=Apfel`)
- `required = false` macht Parameter optional

Quellen

- Spring Boot Documentation
- Building a RESTful Web Service
- Consuming a RESTful Web Service with jQuery
- TASK.md - Aufgabenstellung