Examen: Java Pet Store.

Beoordelingscriteria sourcecode

De student kan softwarecomponenten combineren tot een goed functionerend geheel.
De student kan kwaliteitsvolle applicaties ontwikkelen en implementeren.

Om aan deze vereisten in de studiegids te voldoen verwachten wij van de student:

- compileerbare en werkende code
- respecteer de Code Conventions van de programmeertaal
- code in commentaar wordt niet beschouwd als werkende code

Deze vereisten hebben impact op je eindscore indien niet aan voldaan!

EINDE EXAMEN

Op het einde van het examen voer je de volgende stappen uit:

- Als je klaar bent sluit je je IDE af.
- Maak op je Bureaublad een map <*AchternaamVoornaam_JAVA_ADV2_ZIT1*>. Bijvoorbeeld: *JohnDoe_JAVA_ADV2_ZIT1*.
- Kopieer het project met de oplossing naar de map op je Bureaublad.
- Controleer of al je bronbestanden in de map staan.
- Maak een zip bestand van deze map.
- Bereken de MD5 hash van het zip bestand, en schrijf het resultaat (duidelijk leesbaar) op je examenkopij.

Lever het zip bestand in via FileZilla.

De applicatie

De Java Pet Store is een website waar je een huisdier kan aankopen.
Voor de Java Pet Store website ontwikkel je een RESTful API in Spring Boot. Je start niet “from scratch”, maar je downloadt het project van de ftp-server.
Tijdens dit examen implementeer je enkele user story’s.
Bekijk ook de **README.md** voor extra info!

Gegeven

We maken voor onze implementatie gebruikt van een H2 database. De tabellen worden automatisch aangemaakt wanneer de toepassing wordt opgestart. Tijdens het opstarten wordt ook de code in de klasse ImportData (package be.pxl.petstore.config) uitgevoerd, waardoor data in de databank geladen wordt.

De belangrijkste concepten in het domein-model zijn:

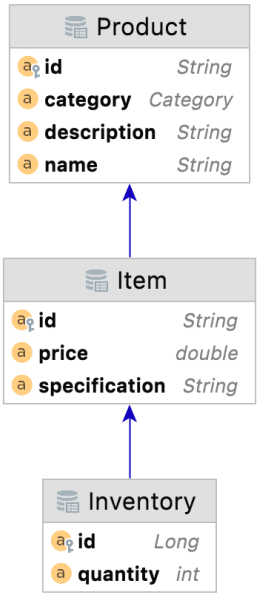
- **Product**: stelt een dier in de pet store voor.
- **Category**: categorie van een product (fish, dogs, cats, reptiles, birds).
- **Item**: concrete details van een product zoals prijs en specificatie. Je hebt bijv. product poodle, maar je hebt de items ‘puppy male’, ‘adult female’, … met verschillende prijzen.
- **Inventory**: voorraad van een item.

De bovenstaande klassen vind je terug in het package be.pxl.petstore.domain.

Gevraagd

☐ Vervolledig eerst de entity klassen Product, Item en Inventory. Er zijn GEEN bi-directionele relaties tussen de entity klassen. (zie ER-diagram)

☐ Voeg de ontbrekende annotaties toe in de klasse PetStoreController (package be.pxl.petstore.rest) zodat het REST endpoint ‘GET /petstore/items’ aangeroepen kan worden (zie ook README.md). Na het toevoegen van de annotaties kan je de toepassing opstarten en het endpoint uittesten.



☐ Implementeer nu onderstaande user story's volgens de opgegeven richtlijnen.

Respecteer de 3-tier architectuur van de toepassing!

Titel: User story 1
Als klant kan ik een bestelling (order) plaatsen in de Pet Store.
Omschrijving: Als klant geef ik mijn naam en email. Een order bestaat verder uit 1 of meerdere items met het aantal. Het moet dus mogelijk zijn meerdere items in 1 keer te bestellen. (zie README.md voor json-formaat van het request)
Acceptatiecriteria: ☐ De naam en email van de klant zijn verplichte velden. ☐ Het opgegeven email adres moet geldig zijn. ☐ Als een item met het opgegeven itemid niet bestaat, gooi je een NotFoundException. Deze exception geeft http-status NOT_FOUND. Het order wordt in dat geval niet opgeslagen. ☐ Als er geen voorraad is van een item in het order, zorg je ervoor dat er een InventoryOutOfStockException wordt gegooid. Deze exception geeft http-status BAD_REQUEST. Het order mag dan niet opgeslagen worden. ☐ Als alle gegevens geldig zijn, wordt het order met zijn orderlijnen correct opgeslagen in de databank. **Testen** (JUnit / Mockito / MockMvc)**:** ☐ Schrijf 1 unit test voor de RestController voor het succesvol plaatsen van een order.

☐ Schrijf 1 unit test voor de RestController waarbij een item niet meer in voorraad is. ☐ Werk 1 relevante unit test uit voor de methode die je implementeert in de service-laag waarbij je mockito op een zinvolle manier gebruikt.
Implementatie-details: Je kan gebruikmaken van de klassen **OrderRequest** en **OrderRequestLine** in het package be.pxl.rest.data om de inkomende json te laten mappen naar objecten. Maak de entity klassen Order en OrderLine met hun relatie(s). Implementeer de gevraagde businessregels, exception-handling en unit testen. **Let op: je moet de databanktabel 'orders' noemen, als je tabelnaam 'order' gebruikt krijg je een foutmelding omdat dit een gereserveerd woord is in SQL.**
REST endpoint: zie README.md

Titel: User story 2
Als beheerder kan ik een overzicht krijgen van alle orders in de Pet Store.
Omschrijving: Voorzie een WebServlet **OF** een thymeleaf-template die een overzicht geeft van alle orders. (URL mag je zelf kiezen)
Acceptatiecriteria: ☐ Een overzichtelijke HTML-pagina met alle orders met bijhorende orderlijnen wordt getoond. ☐ Toon ook de totaalprijs van elk order.
Implementatie-details: Je mag dus kiezen of je gebruikmaakt van een servlet of Thymeleaf. Als je kiest voor een Servlet kan je gebruikmaken van de hulpmethoden in de klasse OrderServlet (package be.pxl.petstore.servlet). Ook de klasse OrderDTO is reeds voorzien (de constructor staat momenteel nog in commentaar omdat je eerst user story 1 moet implementeren).
Voorbeeld html response: Tom Schuyten (tom.schuyten@pxl.be) 1 X Poodle (Adult Female) Total: 145.49 Nele Custers (nele.custers@pxl.be) 3 X Angelfish (Large) 3 X Angelfish (Small) Total: 96.0

We wensen jullie veel succes!