# Opgaven Persistentie

## Voorbereiding.

### Startcode

In het bestand auction.zip vind je een aanzet tot een applicatie die gegevens over een veiling beheert. Deze applicatie gaan we met behulp van JPA annotaties uitbouwen tot een applicatie die deze gegevens beheert in een database. Download het auction.zip bestand van Sharepoint, pak het uit en open het project in Netbeans..

In het package auction.domain vind je de klassen die de gegevens beschrijven waar het over gaat. Er zijn gebruikers (User) die biedingen (Bid) doen op items(Item), die tot een bepaalde categorie (Category) behoren. De gebruikersadministratie is een klein beetje uitgewerkt. In de klasse RegistrationMgr worden zijn de business methoden uitgewerkt. Deze business methoden maken gebruik van de methoden die beschreven worden door de interface auction.DAO.UserDAO. De methoden in de DAO laag zijn fijnmazige methoden die communiceren met de opslag. In de huidige versie wordt de opslag geïmplementeerd door de klasse auction.dao.UserDAOCollectionImpl, die de gebruikers opslaat in een HashMap. (Voor studenten die het vak JCF41 niet gevolgd hebben wordt deze term verklaard in bijlage 1.)

Bestudeer de specificaties (in JavaDoc[[1]](#footnote-1)) en de code van de genoemde klassen en de JUnittest in klasse RegistrationMgrTest. Vergewis je ervan dat alles volgens de specificatie werkt door de test uit te voeren.

## Bestudeer:

|  |
| --- |
| Hoofdstuk 4 |
| Hoofdstuk 5 t/m paragraaf *Access Type* |
| Hoofdstuk 6 t/m hoofdstuk *Entity Manager* |
| De bank applicatie (bank.zip) |

## Opdracht 1

In deze opdracht gaan we de domein objecten persisteren m.b.v. JPA. De objecten worden opgeslagen in een reeds gecreëerde database 'veiling'. Er wordt van uitgegaan dat 'auction' het 'Main Project' is.

1. Annoteer het bestand User.java, zoals dat in de voorbeelden in de boek en de presentatie 'SE42 persistentie 1' is gedaan.
2. Selecteer de Project tab. Kies in het menu **File|New File …** Kies de categorie 'Persistence' en vervolgens 'Persistence Unit' Klik op **Next >.** Kies de goede databaseconnection of maak een nieuwe aan en **Finish**. Je kunt de gegenereerde configuratiefile ‘persistence.xml’op twee manieren bekijken, nl. als XML-tekst en via de Designer. Via de designer kun je bij voorbeeld instellen of de tabellen automatisch gecreëerd en/of gedropt moeten worden. Verder kun je de entity klassen selecteren die gepersisteerd dienen te worden. Zet bij alle opgaven de optie ‘**Shared Cache Mode**’ op **None**!!
3. Voeg via de XML interface in de configuratiefile de volgende regels toe (zie listing 2-5 uit boek):

<property name="eclipselink.logging.logger" value="DefaultLogger"/>

<property name="eclipselink.logging.level" value="FINE"/>

Deze regels zorgen ervoor dat bij voorbeeld gegenereerde SQL statements in de applicatie-console te zien zijn. Handig voor het opsporen van fouten.

1. Maak een kopie van de klasse UserDAOCollectionImpl en noem deze UserDAOJPAImpl. Je gaat deze klasse nu ombouwen zodat niet meer gebruik gemaakt wordt van een HashMap, maar van database persistentie. Pas ook RegistrationMgr aan zodat deze gebruik maakt van de nieuwe service.
2. Kopieer ook de klasse RegistrationMgrTest naar JPARegistrationMgrTest. Pas deze klasse aan zodat deze de klasse RegistrationMgr uit de vorige stap gebruikt, maar laat de testen functioneel hetzelfde doen.
3. Controleer ook of de tabel en inhoud van de database goed worden aangemaakt.**Opmerking 1**: assertSame != assertEquals

In de RegistrationMgrTest wordt de assertSame assertie gebruikt om de ‘gelijkheid’ van objecten te bepalen. In de sheets van week 1 zijn verschillende *identities* van objecten behandeld.

Welke *identity* ga je gebruiken in de JPARegistrationMgrTest om objecten met elkaar te vergelijken, nu entiteiten uit de database geïnstantieerd worden? Bestudeer de asserties assertSame en assertEquals. Welke assertie past bij de gekozen *identity*?

**Opmerking 2**: @Before

Observatie: bij het herhaaldelijk uitvoeren van dezelfde test zijn de resultaten verschillend.

In de test RegistrationMgrTest wordt voor elke test methode de dataset opnieuw geïnitialiseerd door een nieuwe instantie van UserMgr aan te maken. Bij het opslaan van gegevens in een database blijven na afloop van de uitvoering een test methode mogelijk gegevens in de database achter die de volgende testmethode beïnvloeden.

Hint: bestudeer en gebruik de klasse DatabaseCleaner uit de Test Package van de bank applicatie.

**Opmerking 3**: Clean en rebuild regelmatig

Bij het aanpassen van persisitence.xml, entities wellicht ook andere klassen wordt door NetBeans geen adequate automatische clean en rebuild gedaan. Doe dit daarom regelmatig zelf.

## Beoordeling

|  |  |  |
| --- | --- | --- |
| **4** | **7** | **10** |
| Wel programma code geschreven maar meerdere fouten en/of niet werkende en correcte testen. | Enkele kleine fouten in de annotaties, DAO implementatie of testen. | Correcte realisatie van de opdrachten en verantwoording van de gemaakte keuzes. |

1. Javadoc-documentatie kan het best bekeken worden op een HTML-pagina. Klik hiervoor met de rechtermuisknop op het projectknooppunt van auction in het Projectswindow en kies 'Generate Javadoc' in het pop-upmenu. [↑](#footnote-ref-1)