# Kwetter: JPA

Het bijhouden van gebruikers en tweets in collecties op je server stuit tegen grenzen aan. Wegens de stormloop op je website gaan de collecties het geheugen van je computer te boven. Gelukkig is een redeploy nog niet nodig geweest omdat je applicatie uiterst betrouwbaar is. Vooral vanwege het eerste punt dus is het zaak dat objecten in de database worden opgeslagen. Tijdens zijn “JEE FROM A MANAGEMENT PERSPECTIVE”-cursus heeft je baas vernomen dat JPA dè manier is om object-relationele persistentie af te handelen, dus ga je dit gebruiken. Het moet wel mogelijk blijven in verband met regressie-tests om de oude collectie implementatie weer te gebruiken. Je gaat de JPA migratie dus zodanig doen dat, door het aanpassen van de configuratiefile *beans.xml*, dan wel het veranderen van een qualifier op het injectiepunt van de DAO, je applicatie gegevens bijhoudt in een lijst dan wel de database.

De service laag die je gaat bouwen zal na voltooiing bestaan uit de bestanden *UserDAO.java*, *UserDAO\_CollectionImpl.java* en *UserDAO\_JPAImpl.java*. De laatste twee klassen implementeren de interface *UserDAO*. Hieronder wat (onvolledige!) code fragmenten om een en ander te illustreren.

|  |
| --- |
| ***UserDAO\_CollectionImpl*** |
| @Stateless  public class UserDAO\_CollectionImpl implements UserDAO {  private List<User> users; |
| ***UserDAO\_JPAImpl*** |
| @Alternative  @Stateless  public class UserDAO\_JPAImpl implements UserDAO {  @PersistenceContext  private EntityManager em; |
| ***Beans.xml*** |
| <beans …… >  <alternatives>  <class>…... UserDAO\_JPAImpl</class>  </alternatives>  </beans> |
|  |

Nu je twee implementaties hebt van de UserDAO interface kan de applicatie container niet meer automatisch injecteren. De annotatie @Alternative samen met de specificatie in de *Beans.xml* file geven de mogelijkheid om aan te geven welke implementatie de container moet gebruiken. Dit kun je ook realiseren met verschillende qualifiers.

In onderstaand stappenplan wordt nu beschreven hoe je de overgang van lijst opslag naar database opslag kun realiseren. Begin eerst met het bestuderen van hoofdstuk 3, 4 en 6 uit het boek *Beginning JAVA EE 7* (Herhaling van SE42) .

## Stappenplan

1. Prepareer je favoriete database op je laptop: gebruik geen cloud database om traagheid of andere problemen te voorkomen.
2. Annoteer de domain klassen en relaties hiertussen
3. Creëer een bestand persistence.xml. (rechter muis op project → new → other → persistence → persitence unit)
4. Schrijf de klasse *UserDAO\_JPAImpl*.
5. Pas *beans.xml* aan zoals hierboven beschreven, of, maak een andere qualifier aan voor de JPA implementaties van je services.
6. Testen maar. (Wellicht moet je terug naar stap 3)
7. De applicatie werkt weer als voor de refactoring, zowel met de lijstopslag als met de database persistentie.

Opmerkingen

1. Als je er bij de vorige opdrachten voor gekozen had om een TweetDAO aan te maken, moet je voor deze klasse natuurlijk ook een JPA variant schrijven.
2. Wellicht kom je bij stap 7 tot de ontdekking dat de UserDAO handiger gedefinieerd kan worden om het switchen tussen de twee implementaties van UserDAO transparant te maken voor de controller laag en de UI klassen. Voer dan de noodzakelijke wijzigingen in de service -, controller - en UI laag door.

**MoSCoW**

In de berekening van het eindcijfer JEA6, een gewogen gemiddelde van alle opdrachten, wordt deze opdracht 1 keer meegeteld.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Must Have | Should Have | Could Have |
| JPA -entities | X |  |  |
| + Named Queries |  | x |  |
| + Abstract Facade |  |  | x |
| + Efficientie (fetch/eager)  (trends/followers) |  |  | x |

De kwaliteit van de programmacode wordt meegenomen in de beoordeling.