# Kwetter - Rest & CDI

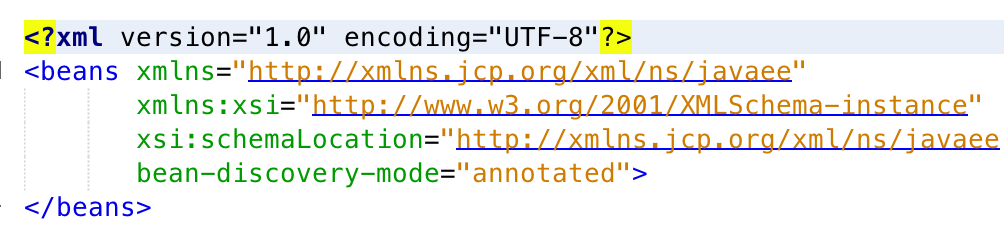
**Inleiding**

In deze opgave ga je de back-end van de kwetter applicatie ontwikkelen. De back-end is een Java EE7 applicatie met een Restfull Web Service voor de communicatie van de data met de front-end. De front-end is een AngularJS applicatie die je in de vorige opgaven hebt ontwikkeld.

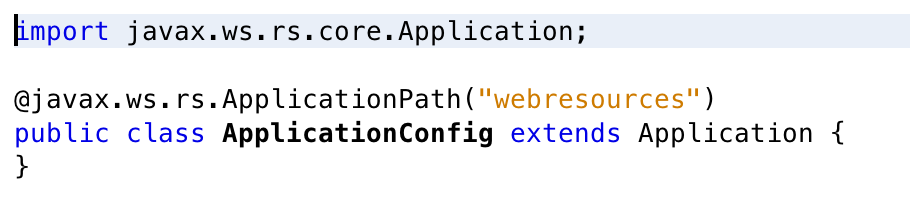
**Project configuratie**

Maak een nieuw “Maven Web Application” project voor de back-end, bijvoorbeeld met de naam Kwetter-Backend. Importeer in het project de source van de benodigde domein en service klassen, die te vinden zijn in het bestand “kwetter.zip” (SharePoint). De domein en service klassen geven een basis die verder ontwikkeld moeten worden.

Maak een nieuwe deployment descriptor (beans.xml) aan in de WEB-INF folder. Belangrijk is dat de eigenschap“bean-discovery-mode” op “all” wordt gezet, dit configureerd hoe CDI met zijn dependency omgaat.



Voor het bouwen van Rest Backend Service moet er gebruik gemaakt worden van JavaEE 7/ JAX-RS. Er is een extra configuratie nodig om JAX-RS te activeren. Neem de onderstaande code op in je project



Hiermee wordt aangegeven dat alle urls die beginnen met “/webresource” RESTful web services zijn.

**Opdrachten**

1. Ontwerp en implementeer een RESTfull Web Service waarmee de users en tweets via XML of JSON kunnen worden aangeboden. Maak een nieuwe resource laag aan boven op de service laag, waarin bijvoorbeeld de klassen KwetterResource de functionaliteit van de RESTfull Service ter beschikking stelt.
2. De Rest EndPoint Services van de back-end is nu klaar om benaderd te worden. In de AngularJS front-end heb je de back-end nog ge-mocked, door in de front-end een AngularJS Service op te nemen. Pas de front-end zodanig aan dat deze gebruik maakt van deze Rest EndPoint Service die bijvoorbeeld de klassen KwetterResource aanbied. In AngularJS kun je gebruik maken van de @Resource service om een REST Service te benaderen. Een voorbeeld hiervan kun je vinden in de [PhoneCat](https://docs.angularjs.org/tutorial) tutorial de 11de stap “[Rest and Custom Services](https://docs.angularjs.org/tutorial/step_11)”
3. Alle interne services (bijvoorbeeld TweetDAO, UserDAO et cetera) dienen als volgt gerefactored te worden.
   1. De service definitie moet in een interface worden geplaats (bijvoorbeeld UserDAO). De Java-conventie is dat het interface een beschrijvende naam heeft: bijvoorbeeld UserDAO (*dus* *niet* IUserService: dit is een .NET conventie).
   2. De implementatie(s) moet/moeten dit interface implementeren. De Java-conventie is dat een implementatie de naam krijgt interface-naam + “Bean” of interface-naam + “Impl”. Specifieke implementaties krijgen een specifiek toevoegsel, bijvoorbeeld Bijvoorbeeld: UserDAO\_CollectionImpl of UserDAO\_JPAImpl voor een toekomstige JPA-implementatie.
   3. Een implementatie moet geïnjecteerd worden in de component die deze service gebruikt. De scope van elke component moet kloppen!
   4. Indien er van een service meerdere implementaties zijn, moet een bepaalde implementatie geïnjecteerd kunnen worden met een specifieke qualifier.
4. Met het oog op de toekomst moet Kwetter een “loosely coupled architecture” krijgen. Dit betekent onder andere dat events in verschillende componenten gepubliceerd en geconsumeerd moeten kunnen worden.

Herontwerp je bestaande Kwetter project zodat de volgende gebeurtenissen door middel van het produceren en consumeren van CDI events worden afgehandeld. Productie en consumptie van een event dient steeds in verschillende componenten plaats te vinden:

* 1. De server ontvangt een tweet van een gebruiker: hiertoe dient een Tweet-event te worden gepubliceerd. Dit event moet onder andere geconsumeerd worden door de component die de tweet opslaat.
  2. Een gebruiker meld zich aan bij je web-applicatie: hiertoe dient een User-event gepubliceerd te worden. Dit event moet geconsumeerd worden door een component die deze gebeurtenis naar het server-log wegschrijft. Het inloggen zelf mag nog steeds gesimuleerd worden.
  3. Zoals de vorige deel-opgave, maar dan voor het afmelden van een gebruiker (bijvoorbeeld doordat zijn sessie beëindigd wordt)
  4. Een kwetteraar begint een andere kwetteraar te volgen.
  5. Een kwetteraar stopt het volgen van een andere kwetteraar.

1. Interceptors: vet en cool zijn uit, hard en dik zijn nu helemaal te dik**.**

Om nieuwe gebruikers van kwetter te helpen om modern en trendy over te komen ga je bij nieuwe tweets de woorden vet of cool automatisch vervangen door dik respectievelijk hard.

Dit doe je door hiervoor een speciale klasse VolgTrendInterceptor te schrijven. Deze klasse neemt de vertaling voor zijn rekening. Zorg er ook voor dat deze interceptor bij de juiste methode aangeroepen (invoked) wordt.

**MoSCoW**

In de berekening van het eindcijfer JEA6, een gewogen gemiddelde van alle opdrachten, wordt deze opdracht 4 keer meegeteld. De kwaliteit van de programmacode wordt meegenomen in de beoordeling.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Must Have | Should Have | Could Have |
| Opdracht 1 | X |  |  |
| Opdracht 2 | x |  |  |
| Opdracht 3 | X |  |  |
| Opdracht 4 |  | X |  |
| Opdracht 5 |  |  | X |