URV GEI 2023-2024

Sistemes Oberts: Práctica 1

Web Backend with Jakarta and API REST

Jialiang Chen, Ivan Garcia

02-12-2023

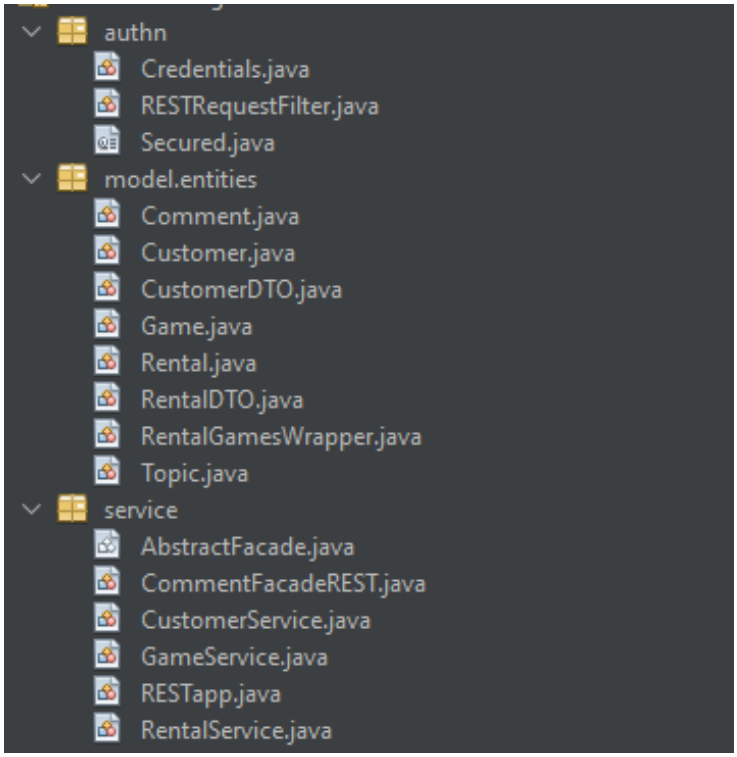
Presentació de la práctica

En aquesta práctica l'objectiu principal és aprendre a utilitzar I implementar un backend d'una palicació web usant Jakarta Persistance, glassfish I API REST. No és necessari fer res de front-end en aquesta pràctica.

Per començar amb una ajuda, es proporciona un esquelet buit d'una aplicació web en forma de projecte de NetBeans19.

Estructura de la pràctica

L'estructura de la pràctica és la següent:



El package authn l'he deixat com està, ja que està implementat I s'usa per poder usar autenticació per les Queries.

El package entities té totes les classes d'entitat I DTOs (Data Transfer Object), principalment Game, Rental I Customer, dels quals Rental I Customer tenen una clase DTO per poder enmascarar les dades de les classes I mostrarles com a resultat d'una queri.

La classe RentalWrapper s'usa per poder rebre amb una query un objecte java Rental I una llista d'IDs de Game, en format JSON, per exemple el següent:

Decisions de disseny:

Per fer les queries, principalment hem fet servir les NamedQueries personalitzades a cada entitat, ja que són més eficients I es compilen a temps de compilació. A més hem aprofitat la clase AbstractFacade per fer algunes de les operacions sobre la Entity Manager com create, edit o persist.

Pel que fa les IDs de cada entitat, hem decidit fer servir l'autogeneració en seqüència amb unitats d'al·locació 1, ja que en aquesta pràctica no es farà servir moltes instàncies d'entitats, no cal un generador més eficient, I com que no fem càrregues massives de dades, només cal separar en 1 unitat cada ID.

En l'entitat Game, hem decidit que l'adreça es guardi en una string, I que quan es modifiqui s'agafi el valor antic I s'afegeixi el nou al darrere. A més, hem usat el tipus ENUM per guardar el gènere I consola de l'entitat, per tenir un control sobre quins tipus de gènere o consoles hi han.

Pel que fa la relació entre Rental I Game, hem usat una relació many to many per poder guardar n jocs en n alquilers, per fer possible aquesta relació s'ha creat la joined table "RENTAL_GAME" per poder guardar la relació, això vol dir que a game I rental no es guardarà cap referència a nivel de Base de Dades, encara que en Java, sí.

La majoria de casos que hem tingut que usar llistas, hem usat una List<T> per simplicitat.
A totes les entitats se li afegeix el prefix /rest/api/v1/ a la path (requeriment de la pràctica).

S'usa les PathParam per poder agafar un paràmetre en la path de la URL, o QueryParam si es vol agafar un paràmetre de la URL (notació ?x=x&y=y).

S'anota amb @Produces I @Consumes per poder fer la traducció automàtica de JSON/XML a objecte I viceversa.

Es fa inserts de la base de dades en l'arxiu install.jsp per tenir una base de dades bàsic per provar (veure l'arxiu per més detalls).

GameService:

- **POST**: En el post usem una query per trobar si hi ha un job amb les mateixes característiques (per això necessitem tants paràmetres), I en cas de que existeixi retornem un CONFLICT. Per crear el Game en cas de que no existeixi es crearà un usant el mètode de la classe pare super.create(game);.
- **GET**: Al principi de tot es valida si genre I consola tenen un tipus vàlid (que es pugui convertir de String a les ENUMs corresponents, en cas de que no es pugui es farà el "catch" de l'excepció I es retorna BAD_REQUEST). En cas de que genre I console siguin vàlids (o que no s'hagi passat els paràmetres), es fa la query segons si:
    - Hi ha genre pero no console
    - Hi ha genre I console
    - No hi ha genre pero si console
    - No hi ha genre I tampoc console (en aquest cas es retorna tots els jocs).
    - Hi han els dos paràmetres

  Per poder retornar el resultat en forma ordenada es fa servir la sentència SQL ORDER BY ASC.

CustomerService:

- **GET (pels dos mètodes)**: Per poder retornar l'informació sense informació confidencial s'usa la clase CustomerDTO (Que només té les dades no confidencials) per poder mostrarles amb seguritat. En el cas de get amb id, es fa una NamedQuery simple per poder trobar el Customer.

- **PUT**: Es fa una query simple per veure si ja existeix el customer o no, Si ja existeix, es modifica amb super.edit, si no existeix, s'usa em.persist().

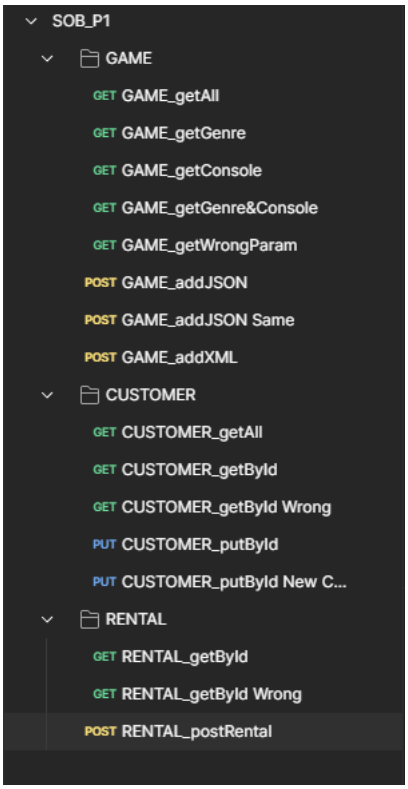
RentalService:

- **GET**: S'usa la clase DTO corresponent per respondre només l'informació demanada, en cas de que no existeixi es retorna NOT_FOUND.
- **POST**: Primer es crea les variables auxiliars necessaris per poder comprobar que tots els jocs existeixi I guardar els resultats de les queries. Després es comprova que el customer existeixi, sino es retorna NOT_FOUND.

  En un for es fa la comprovació de que els jocs existeixin, si no es així es retorna un NOT_FOUND. En cada iteració s'actualitza el Game amb una nou rental (addRental) I s'actualitza rental amb un now Game (addGame). Al final es fa el set de la llista de jocs de rental I es fa un merge de rental (ja que s'ha modificat les dades).

Jocs de proves

Per tal de comprovar que funcioni el nostre backend, hem usat l'eina PostMan per crear queries personalitzades I testejar cadascun de les APIs per separat, hem creat 3 carpetes per cadascuna de les entitats:



PostMan ens deixa còrrer totes les queries en un sol click, que ho fa molt últl a l'hora de debugar o testejar, també es pot exportar les queries que hem creat si el professor el vol fer servir.

Implementació de funcionalitats

Hem implementat totes les APIs obligatoris I **també** les opcionals.

Conclusions

La pràctica ha estat força entretinguda I hem après molt sobre la programació web backend la sincronització de bases de dades entre l'aplicació I les dades passades per la pàgina web (API), encara que una mica difícil al principi de saber com funciona i com implementar les funcionalitats més bàsiques. El servidor glassfish no va molt bé I es podria utilitzar alternatives per fer aquesta pràctica com Spring web.

Manual d'instal·lació

Només cal copiar les nostres clases sobre un esquelet de la pràctica, muntar la BD (per default) I encendre el servidor glassfish. Després cal encendre l'aplicació (apareixerà la pàgina web per default amb el botó install), cal prémer el botó install per fer totes les Insert a la BD per fer els tests. Per provar-lo (les APIs) es pot usar el postman (importar l'arxiu JSON proporcionat) I fer les queries manualment o correr-les totes a la vegada.