Aplicaciones Distribuidas (AD)

Práctica 4: Desarrollo de servicios web con REST

Mònica Benet
Nicolas Muscolo
Samuel Salvador Roca-Cusachs

INDEX

INTRODUCCIÓ

Un cop més hem hagut d'adequar les pràctiques anteriors a una nova pràctica, tot i que aquest cop hem creat els serveis amb Web RESTful. L'entorn segueix sent el mateix que en les darreres pràctiques però el terme REST (Representational State Transfer) ens aporta un conjunt de restriccions amb les quals podem crear un estil d'arquitectura software per a crear aplicacions web. REST és l'alternativa senzilla als altres protocols d'intercanvi de dades com SOAP.

API REST és l'opció més utilitzada en la major part d'empreses. Empreses com Facebook o Google obten per utilitzar-lo per als sistemes d'identificació o autenticació ja que és l'estàndard lògic i eficient per a la creació de serveis web.

Les restriccions que defineixen un sistema RESTful són les següents:

- Protocol client/servidor: manté desacoplats els dos extrems, permetent que el client no conegui els detalls del servidor i viceversa. Cada petició HTTP conté tota la informació necessària per a ser executada, implicant que ni el client ni el servidor hagin de recordar cap estat previ per a satisfer-la.
- Sense estat: cada petició que rep el servidor hauria de ser independent. Per això mateix no es poden mantenir sessions en la part del servidor.
- Interfície uniforme: cada recurs del servei REST ha de tenir una única direcció URI implicant separació en l'arquitectura.
- Sistema de capes: arquitectura jeràrquica entre els components. Cada capa du a terme una funcionalitat dins del sistema.

FUNCIONAMENT DE LA PRÀCTICA

En aquesta pràctica tenim una aplicació web client amb tots els JSP que teníem en la pràctica 2, que es comunica mitjançant operacions GET i POST amb el servidor Web REST. Aquest servidor és qui es comunica amb les classes d'accés a la BD (OurDao.java) que finalment són les que es comuniquen amb la Base de Dades.

Base de dades

Per aquest projecte estem utilitzant la mateixa base de dades que la pràctica anterior, *JavaDB*. Així doncs només ens ha calgut importar el DAO de la pràctica anterior (OurDao.java) amb les funcionalitats gestores de la BD.

Servei REST

Al servei web trobem les següents funcions:

RegistrerImage:

Funció que rep els atributs d'una imatge com paràmetres i fa una crida al DAO que s'encarrega de fer la gestió corresponent de la base de dades, registrant-la a aquesta.

ModifyImage

Aquest mètode rep els paràmetres d'una imatge i els envia al DAO que és l'encarregat de fer els canvis en la imatge corresponent de la base de dades.

ListImages

Funció que mitjançant el mètode *getAllImages()* fa una consulta a la base de dades a través del DAO i retorna una llista amb les imatges de la base de dades. Genera una taula que conté les dades necessàries per a que el client gestioni la sessió (s'afegeix l'id) i altres dades informatives.

SearchById / SearchByTitle / SearchByAuthor / SearchByCreaDate / SearchByKeywords

Aquestes funcions han estat unificades en una anomenada *searchCombi(..)* que rep els paràmetres per part de l'usuari (a través del buscarImagen.jsp). Amb aquests paràmetres fa una cerca pels camps pertinents unificant els resultats en una sola variable que és retornada al .jsp on es mostra a l'usuari.

DeleteImage

Esborra la imatge rebent una imatge com a paràmetre i esborrant-la de la Base de dades a partir de l'id.

registerUser

Aquesta funció permet registrar usuaris nous a la base de dades. També serveix per validar els noms i les contrasenyes a la part del servidor de forma que es validen tant als clients com en aquesta funció. Torna un JSON per indicar si la crida ha sigut exitosa i amb un missatge informatiu adicional.

loginUser

Aquesta funció comprova si un usuari existeix a la base de dades. Torna un JSON per indicar si la crida ha sigut exitosa i amb un missatge informatiu addicional.

getUsers

Aquesta funció serveix per obtenir un llistat de tots els usuaris a la base de dades. El resultat s'envia en format JSON (Array)

Aplicació web client

L'aplicació web client està formada pels JSP que teníem a la pràctica 2 però amb certes modificacions per tal d'adequar-la a les restriccions imposades pel servei REST.

Modificacions

Per tal d'aconseguir que el client sigui totalment independent del servidor, i aquest últim no retorni directament a l'usuaria, hem hagut d'afegir codi Javascript que s'ocupa d'enviar les peticions del client al servidor i rebre les contestacions del servidor per retornar-les al client.

Hem modificat tots els JSP per tal de que hi hagi separació en l'arquitectura de la següent manera: A cada JSP on apareixen formularis hem afegit funcions en JavaScript asíncrones que es comuniquen amb el servei REST fent crides mitjançant la API fetch que ve inclosa. Per a cridar manualment passant els paràmetres dels formularis amb tipus 'application/x-www-form-urlencoded' utilitzem el objecte URLSearchParams que permet crear el tipus de dades que necessitem de manera fàcil. Utilitzem await per esperar el retorn de les crides i processem el JSON amb una crida a *.json()*. Als llocs on es retorna HTML l'incrustem a la propia página per mostrar el resultat.

Gestió de sessions

Tal i com hem indicat a la introducció, la gestió de sessions no es pot fer a la part del servidor per això mateix les hem inclòs en la part del client.
Afegint un script als JSP amb Javascript hem codificat el següent.

```
let ses = window.sessionStorage;
if (ses.getItem('user') === null)
        window.location.replace('menu.jsp');
```

Això també ens ha permès poder enviar valors entre els JSP per tal de saber quin és el ID de la imatge seleccionada a l'hora de modificar o eliminar imatges.

TREBALL ADDICIONAL

Per millorar les funcionalitats de la pràctica hem afegit les següents ampliacions:

- Cerca combinada:
A l'aplicació web hem afegit una funcionalitat que permet a la usuària no només fer una cerca per un sol paràmetre sinó pels que ella decideixi.
Analitzant el seu funcionament, es barregen totes les cerques i es fa servir una sola, la combinada. Aleshores la usuària pot consultar els camps que decideixi, aquests s'envien al servei REST on s'encarrega de cridar individualment segons els seus paràmetres a les diferents cerques i retorna un resultat unificat, una taula html que es ensenyada a la usuària a través del .jsp. S'han evitat les repeticions guardant les cerques individuals en una col·lecció (*HashSet*) per posteriorment agafar aquests resultats i enviar-los a l'usuari.

QÜESTIONS

a) Perquè podem accedir al servei des del navegador?
   Perquè el servei disposa de mètode *get(),* el qual ens permet accedir pel navegador.

b) Què mètode HTTP s'està utilitzant?
   El mètode *get()*.

c) Amb quin tipus MIME?
   L'introduït a la posada en marxa de la pràctica, text/html.

d) On i com s'indica la URL, el mètode HTTP i el tipus MIME del servei REST?
   URL: https://domini:port/ruta-del-recurs?consulta

HTTP: @GET/@PUT

MIME: @Produces(MediaType.TEXT_HTML)

CONCLUSIÓ

Per decisió conjunta, aquesta implementació és la que més ens ha agradat. Encara que haguéssim estimat tenir més temps per desenvolupar-la, considerem que és l'arquitectura més còmode d'entendre, que permet la ràpida escalabilitat i la possibilitat d'afegir noves funcionalitats sense haver de modificar la resta d'estructura.

REFERÈNCIES

Netbeans.org. 2020. *Netbeans*. [online] Available at: <http://netbeans.org/>

Codejava.net. 2020. *Codejava.Net - Java Tutorials, Code Examples And Sample Projects*. [online] Available at: <https://www.codejava.net/>.

W3schools.com. 2020. *W3schools Online Web Tutorials*. [online] Available at: <https://www.w3schools.com/>.

Oracle Help Center. 2020. *JDK 15 Documentation - Home*. [online] Available at: <https://docs.oracle.com/javase>.