Software Distribuït

Pràctica 2: Llibreria de Recursos Electrònics Online.

Vicent Roig / Igor Dzinka

01/06/2015

Índex

1. Objectiu

L'objectiu docent de la pràctica és aprendre a utilitzar els mecanismes de programació Client/Servidor en JAVA. Concretament és necessari que aprengueu com programar amb:

- Sockets amb JAVA (utilitzant l'API Socket de Java.net)
- Servidor multi-petició amb i sense threads (JAVA)

2. Introducció

Es desitja fer una llibreria Online de recursos digitals, que permeti als usuaris baixar-se llibres, cançons, vídeos curts... en qualsevol tipus de format mp3, pdf, avi... si tenen prou crèdit per adquirir-les.

1. Programació en llenguatge Java usant Servlets i JSP.
2. Ús de Tomcat com a contenidor de servlets.
3. L'autenticació usarà l'especificació JASS.
4. Implementació d'un Servei Web RESTful usant servlets i JSON.

3. Descripció detallada de la implementació

3.1 Resum de característiques:

L'aplicació web esta formada per un catàleg de diferents recursos digitals modelats a una classe **JavaBean** anomenada '**Product'**. Aquesta classe compta amb un atribut FileType que classifica el tipus de producte en qüestió. Els tipus de productes segons especificacions de l'enunciat són:

- Llibres: `FileType.BOOK`
- Cançons: `FileType.AUDIO`
- vídeos curts: `FileType.VIDEO`

La classe DataManager ubicada al Package controller, proporcionarà els mecanismes de tractament de dades necessaris per a que el ServletDispatcher (controlador *"mestre"* de l'aplicació) disposi de les estructures dels productes i usuaris enregistrats a la part del servidor. Per a carregar aquestes dades s'ha fet ús de col·leccions trhead-safe que permeten concurrència per a múltiples fils. En particular, s'utilitza la estructura ConcurrentHashMap utilitzant com a Key el que em decidit anomenar PID (Product Identifier) al cas dels productes, o el username/name al cas dels usuaris.

Observem que el procés de construcció i parseig dels JSON es realitza via la llibreria de google GSON, la qual ens permet obtenir fàcilment la representació JSON d'un JavaBean de forma directe si aquesta segueix correctament la definició de Bean (Implementar ***Serializable***, constructor sense parametritzar, atributs privats, setters/getters públics...).

El ServletDispatcher instanciarà la classe DataManager al mètode init() per tal de carregar les dades correctament, si és necessari afegir un usuari no existent, ens porporciona el mètode addUser, entre altres. Finalment cada cop que s'aturi el Servlet, al procés onDestroy es desaran les dades dels usuaris per tal de mantenir integritat i persistència.

Aquesta classe és en definitiva el Handler de les dades i proporciona al controlador la capa d'abstracció necessària per al controlador. Segueix el patró Singleton, per tant un cop ha passat per l'init(), les dades carregades seran úniques i forman part d'una sola instància de classe.

DataManager defineix alguns dels comportaments més interesants que utilitzarem al projecte, com ara el tipus de Producte (obviem l'herencia múltiple sobre una classe abstracte Product/Item), mètodes encarregats de la persistència com ara saveUsers, i finalment dos mètodes estàtics que ens permetran realitzar consultes sobre les dades als propis JSP via scriplets sense tenir que instanciar-la de nou:

- `public static ConcurrentHashMap<String, Product> getProducts()`
- `public static ConcurrentHashMap<String, User> getUsers()`

3.2 Altres característiques:

- Sistema d'autentificació JASS, seguint l'exemple facilitat al material docent del CV.
- "Llista de descàrregues" per a que l'usuari de la llibreria pugui anar-hi ficant tots els recursos que vulgui baixar-se.
- Un cop hagi seleccionat tot el que vulgui, podrà descarregar-se els recursos des de la pàgina de descàrregues, mentre li quedi crèdit.
- El crèdit inicial de cada usuari estarà fixat per l'aplicació web (500€ cada cop que es requereix el registre d'un nou usuari).

3.3 Estructuració del projecte:

Al projecte s'ha seguit el patró **Model-Vista-Controlador**, la estructuració de classes segueix la següent distribució:

En quant als fitxers, a nivell jeràrquic, cadascun dels tres projectes Java les classes s'han distribuït en 3-4 paquets en funció de paper que hi juguen. Òbviament els principals paquets segueixen el patró de l'arquitectura MVC:

- **/controller** on trobarem com a **Controladors** el `ServletDispatcher` (WebApp) junt al `WebServiceServlet` (controlador delegat a la part del WebService), junt a la classe `DataManager`. Hem decidit situar aquí aquesta classe perquè implementa parts pròpies a un controlador com ara és guardar al fitxer i carregar les dades.

- **/model** Emmagatzema la classe `User` la qual conta amb els atributs i estructures necessàries per mantenir la informació del usuari en temps d'execució. (Ex: `ArrayList purchased, ArrayList cart` per als productes comprats i actualment al carret corresponentment).

- **/beans** Paquet propi per les classes de model que segueixin la definició estricta de JavaBean.

Com ja s'ha comentat prèviament a nivell de disseny, DataManager interessa que sigui d'única instanciació i per tant, implementa el patró **Singleton.**

Per altra banda, s'han definit dos paquets a la capa de model, diferenciant les classes que segueixen la definició de JavaBean en el sentit més estricte d'aquelles que pel propi disseny no es convenia que fos així, notem que al nostre cas User ha de mantenir un constructor propi que permeti construir-ho amb una estructura ArrayList per tant d'agilitzar la consulta de Products vinculats a la conta (purchased/cart) i per tant no obtindríem cap profit de definir-

ho com a JavaBean. En canvi, la classe Product sí ens ho permet i això ens dona la facilitat de treure màxim rendiment a la llibreria GSON a l'hora de parsejar-ho com JSON Object.

Seguint la estructura de projecte de la passada entrega, La versió més actual de la llibreria GSON és referenciada desde la carpeta **/lib** ubicada a l'arrel del repositori.

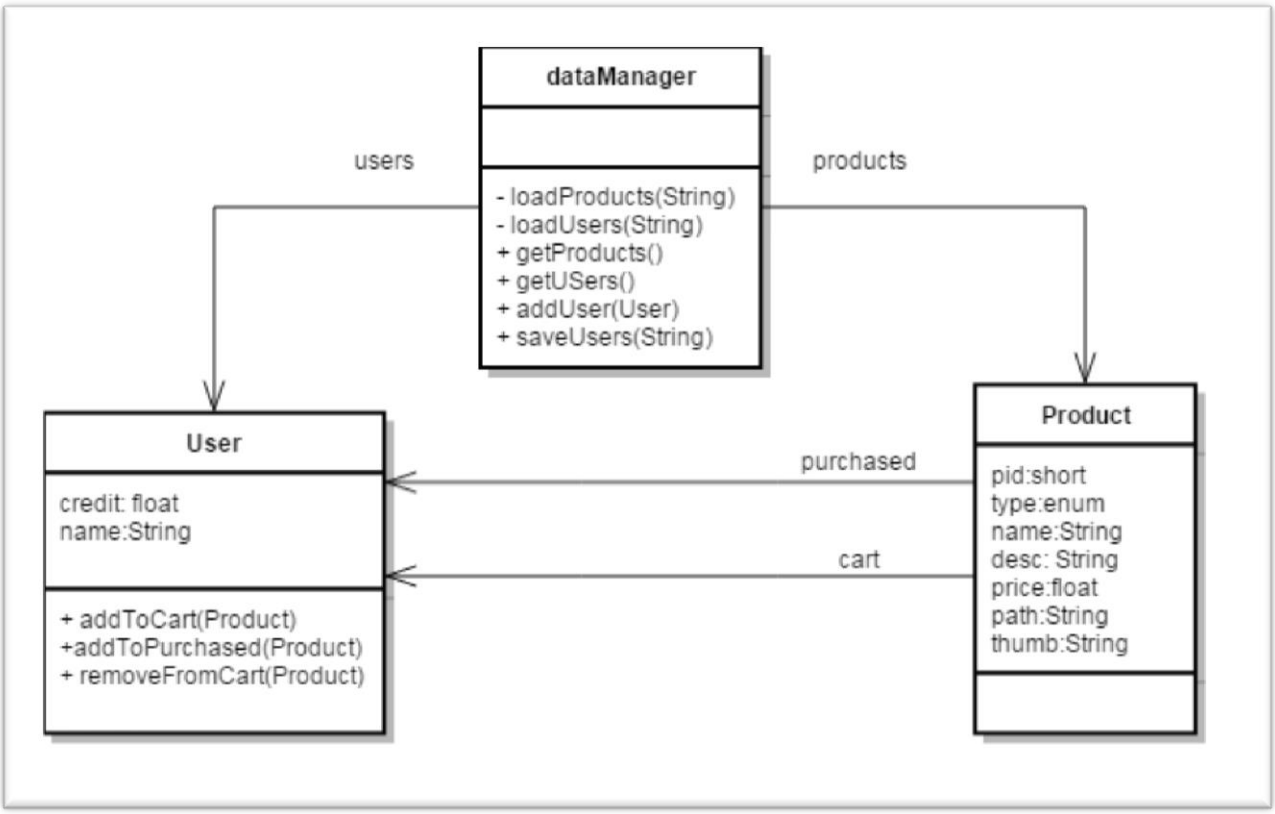
3.2 Format i persistència de les dades:

Tant les dades dels productes com dels usuaris són desats a disc, en concret als fitxers **users.json** i **products.json** ubicats a la ruta de projecte llibreria/web/WEB-INF. El format de les dades és el següent:

```
{
    "products"  :   [
        {
            "pid"    :   1,
            "name"  :   "You",
            "desc"  :   "Nils Frahm",
            "price" :   1.99,
            "type"  :   "AUDIO",
            "path"  :   "/resources/audio/You.mp3",
            "thumb" :   "/llibreria/static/img/thumb_you.jpg"
        },
        {
            "pid"    :   2,
            "name"  :   "Villancico linuxero",
            "desc"  :   "Pánico en el nucleo",
            "price" :   0.99,
            "type"  :   "AUDIO",
            "path"  :   "/resources/audio/villancicolinuxero.mp3",
            "thumb" :   "/llibreria/static/img/thumb_linuxero.jpg"
        }, (...)
```

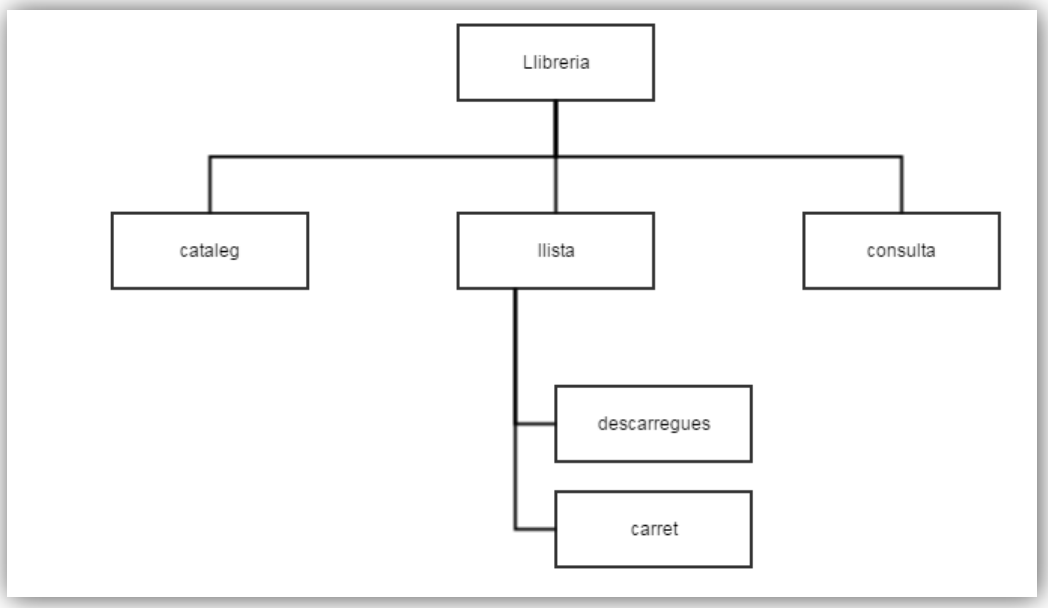
Aquestes dades es carreguen al `init()`, el fitxer **users.json** és actualitzat cada cop que es realitza una crida al mètode `onDestroy()`.

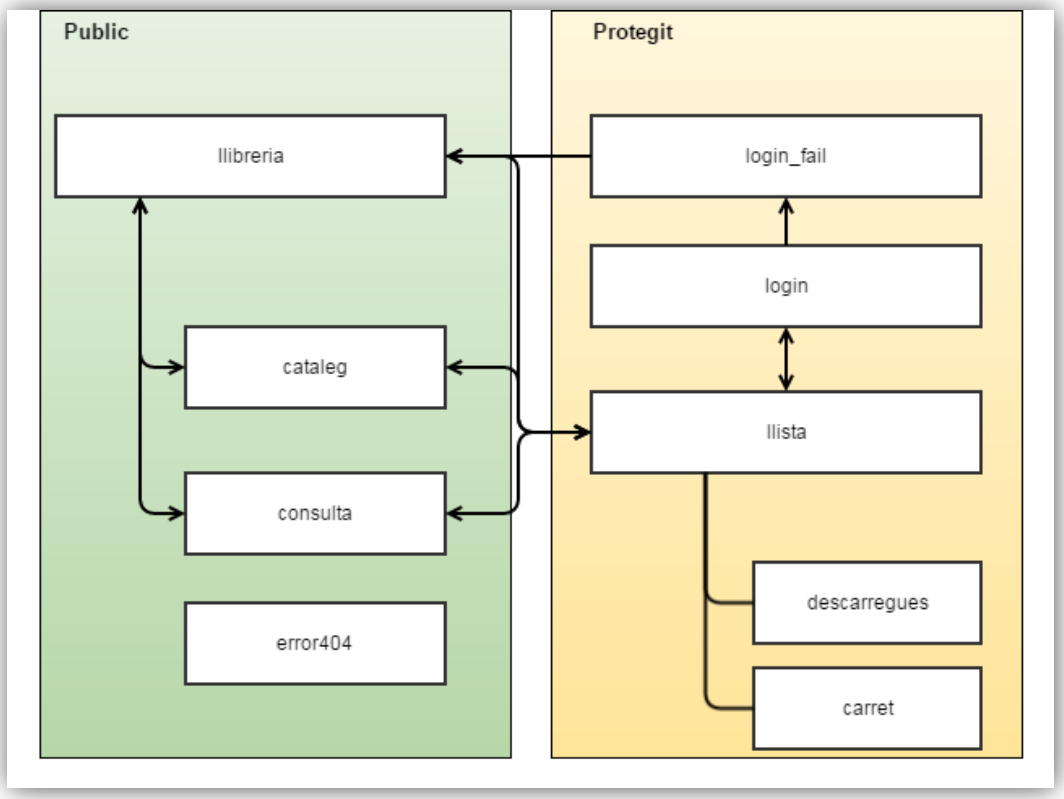
4.Diagrama de Classes



.

5.Mapa web


Llibreria
cataleg
llista
consulta
descarregues
carret


Public
Protegit
llibreria
login_fail
login
cataleg
llista
consulta
error404
descarregues
carret

5. Proves Realitzades

De les proves realitzades als altres grups a la sessió de testing obtenim els següents resultats:

B8:

```
- Logar-se amb usuari existent.                                                    OK
- Logar-se amb usuari no existent.                                                 Error de login OK
- Es disposa d'un limit de diners?                                                 SI
- Realitzar una compra d'un item.                                                  OK
-Intentar tornar a comprar el mateix ítem. (delogar-se primer i tornar a logar)    Es controla
- Veure històric de compres.                                  Es veuen articles i es pot descarregar
- Esborrar cokies, deshabilitar-les en el navegador i tornar a provar.             OK
- Utilitzar el WebService i buscar el producte: BOOK :  Distributed _Systems       -
- Fer servir el link de la llibreria més barata i provar de descarreg ar-lo.       -
- Anar a la llibreria, comprar-lo i tornar a provar el WebService.                 -
- Observacions: --
```

B5:

```
- Logar-se amb usuari existent.                                                    OK
- Logar-se amb usuari no existent.                                                 Error de login OK
- Es disposa d'un limit de diners?                                                 SI
- Realitzar una compra d'un item.                                                  OK
-Intentar tornar a comprar el mateix ítem. (delogar-se primer i tornar a logar)    Es controla
- Veure històric de compres.                                  Es veuen articles i es pot descarregar
- Esborrar cokies, deshabilitar-les en el navegador i tornar a provar.             OK
- Utilitzar el WebService i buscar el producte: BOOK :  Distributed _Systems       -
- Fer servir el link de la llibreria més barata i provar de descarreg ar-lo.       -
- Anar a la llibreria, comprar-lo i tornar a provar el WebService.                 -
- Observacions: --
```

B4:

- Logar-se amb usuari existent.	OK
- Logar-se amb usuari no existent.	Error de login OK
- Es disposa d'un limit de diners?	SI
- Realitzar una compra d'un item.	OK
-Intentar tornar a comprar el mateix ítem. (delogar-se primer i tornar a logar)	Es controla
- Veure històric de compres.	Es veuen articles i es pot descarregar
- Esborrar cokies, deshabilitar-les en el navegador i tornar a provar.	No loguea
- Utilitzar el WebService i buscar el producte: BOOK : Distributed _Systems -	OK
- Fer servir el link de la llibreria més barata i provar de descarreg ar-lo. -	-
- Anar a la llibreria, comprar-lo i tornar a provar el WebService.	-
- Observacions: --	