Scopul acestui laborator este sa realizam un micro magazin virtual.

Vom avea o pagina de login (fara parola, doar un user).

Apoi, odata inregistrat, userul va putea sa isi aleaga niste produse si sa le puna in cos.

Cosul poate apoi fi vizualizat printr-un webservice.

1. Copiati folderul adus de mine pe USB la voi in computer
2. Lansati IDEul preferat (aveti un eclipse in folderul de la mine).
3. Creati un nou proiect cu numele vostru (JavaWeb in netbeans, DynamicWebApplication in Eclipse)
4. Intrati pe disc si modificati fisierul index.html in index.jsp. Daca nu exista, creati index.jsp. Acesta e fisierul de pornire in aplicatie.
5. In JSP putem introduce HTML simplu si prin HTML putem adauga marcaje speciale pentru a introduce cod Java. Vedeti detalii la: <https://www.tutorialspoint.com/jsp/index.htm> (dupa laborator, nu acum).
6. Identificati fisierul index.jsp din folderul de la mine. Deschideti acel fisier si copiati formularul de username si parola de acolo. Doar formularul!!! Faceti build al aplicatiei.
7. Porniti tomcat la voi pe computer. Folositi launcherul din folderul de la mine. Faceti deploy la aplicatie si vedeti cum arata. Corectati eventualele greseli.
8. Adaugati verificare daca userul este logat sau nu. Daca este logat, trebuie sa se afiseze “bine ai revenit NumeUser” daca nu, se afiseaza formularul. Adaugati conditia pe variabila de sesiune, vedeti in modelul de la mine. Copiati codul de acolo si apoi modificati astfel incat sa nu faca decat ce se cere la acest punct. Build si deploy pe tomcat.
9. Cand dam login, primim o eroare. Servletul care face login nu exista. Creati un servlet cu numele Login. Atentie la deployment descriptor! Incercati sa il faceti singuri/cu ajutorul IDEului, doar daca nu va iese, uitati-va in fisierul de la mine. Copiati din fisier doar daca/ce este necesar! In servletul de login, verificati daca a fost introdus un nume de user. Daca da, creeati un cos si pregatiti sesiunea (puneti nueme userului in sesiune). Puteti sa va inspirati din modelul de la mine. NU copiati ultimele doua linii din modelul de la mine! Build si deploy.
10. Acum cand dam login, ne duce intr-o pagina alba. Daca revenim manual in index.jsp, pare sa fie inregistrat userul. De ce? Ganditi-va….incercati sa va raspundeti apoi cititi mai departe…….Adaugati ultimele doua linii din modelul de la mine. In acest fel, servletul se va executa si apoi requestul va fi pasat mai departe catre JSP, iar rezultatul va fi cel produs de JSP (un fel de redirect, dar este intern, browserul userului nu stie ca s-a facut).
11. Haideti sa adaugam produse in cos. Adaugati formularul de “Adauga in cos”. “Inspirati-va” din modelul de la mine. Build si deploy. Primim eroare cand adaugam ceva in cos. De ce? Corectati eroarea si apoi mergeti la 12.
12. Trebuie adaugat cod pentru a pune obiectele in cos. Editati singuri servletul Adauga. Incercati sa va descurcati singuri, este java simplu si cateva din elementele cu care am lucrat pana acum (in acest la laborator). Nu uitati de RequestDispatcher.forward, pentru ca sa ne intoarcem in index.jsp. DOAR DACA NU VA DESCURCATI, inspirati-va din codul de la mine.
13. Haideti sa adaugam lista de obiecte din cos pe site in index.jsp. Ar trebui sa va puteti descurca singuri. Folositi o lista neordonata (<ul> in html) si apoi o bucla care trece prin toate elementele din cos si adauga pentru fiecare un item in lista (<li> in html). Incercati singuri si doar daca nu va descurcati, inspirati-va de la mine.
14. Haideti sa implementam un webservice, mai precis un REST service. Serviciile REST sunt in principiu niste “comenzi” pe care le dam serverului prin intermediul unui URL. Sunt sigur ca le veti gasi foarte utile. In acest caz, am vrea ca atunci cand userul acceseaza URLul /myApp/rest/export/numarObiecte sa primeasca o lista cu obiectele din cosul sau, dar nu mai mult de “numarObiecte” intrari. Pentru asta ne trebuie o librarie speciala si o sa o facem in mai multi pasi. Incercati sa urmariti aici, si NU va uitati pe fisierul de la mine!
15. Adaugati in classpath libraria jersey-bundle-1.19.1.jar din folderul de la mine.
16. Adaugati in WEB-INF/lib acelasi jar (serverul trebuie sa stie de el, sa aiba acces la el).
17. Adaugati un nou pachet cu un nume sugestiv
18. Adaugati o noua clasa ina cest pachet. Aceasta clasa va juca rolul unui servlet. Numele nu conteaza dar alegeti ceva sugestiv.
19. Adaugati adnotarea @Path cu parametrul “/”. Aceasta este calea de mapare la nivel de clasa pentru webservice.
20. Adaugati o metoda publica ce returneaza javax.ws.rs.core.Response, orice nume, si are ca parametru un HttpServletRequest si un Integer maxItems.
21. Pentru ca jersey sa stie sa apeleze cum trebuie aceasta metoda, trebuie sa mai adaugam niste adnotari. Adaugati @GET la metoda. Astfel stim ca trebuie sa servim requesturi de tip GET.
22. Adaugati @Path(“/export/{maxItems}”) la metoda. Astfel instruim serverul ca orice url de forma rest/export/12345 va fi procesat de metoda aceasta iar ce se afla dupa export este un parametru ce va fi pasat functiei. Pot fi mai multi parametrii, de exemplu /export/{param1}/{param2}.
23. Mai departe, trebuie sa specificam ca request trebuie populat din contextul servletului de unde va fi apelata metoda. Adaugati @Context la parametrul request.
24. Acum trebuie sa instruim jersey sa puna valoarea lui maxItems din url. Adaugati adnotarea @PathParam(“maxItems”) la parametrul maxItems.
25. Pentru a construi un Response, folositi builderul static din clasa Response. Ex: Response.status(200).entity(“stringul care va fi raspunsul”).build(); Incercati sa va descurcati singuri si construiti raspunsul asa cum trebuie.
26. Mai trebuie facute niste modificari in deployment descriptor. Aceasta librarie functioneaza generand un servlet care stie sa se uite intr-un anumit pachet si sa caute clasele adnotate corect acolo. Definirea servletului si url-patternul este clasica, dar trebuie sa mai adaugati niste parametrii. Vedeti web.xml facut de mine.
27. Build si deploy. Daca aveti probleme, uitati-va in modelul de la mine. Atentie, cele mai mici greseli pot rezulta in….lipsa unui rezultat :)

Model:

<https://drive.google.com/open?id=0B5ar2tHw-X9vT1M0TDE2ZWZPazA>