Moise Andrei

432A

Programarea aplicativa A interfetelor

ETTI-UPB

Cuprins

1. Introducere…………………………………………2
2. JavaServer Pages……………………………………………..6
3. Prezentarea codului……………………………………………11
4. Bibliografie……………………………………….18

CAPITOLUL 1

INTRODUCERE

* 1. Despre Proiectarea Interfetelor

Cand vine vorba de dezvoltare software, partea cea mai dificila, si cea mai consumatoare de timp este proiectarea si programarea interfetei cu utilizatorul(UI). Potrivit unor studii recente, programarea interfetei utilizator reprezinta undeva intre 30 si 90% din totalul liniilor de cod scrise pentru un software.[1]

Prima si probabil, cea mai importanta regula a proiectarii unei interfete este consistenta. Consistenta reprezinta una dintre cele mai de baza asteptari ale utilizatorului final, iar lipsa consistentei rezulta doar in frustrare in randul utilizatorilor care vor da click ca nebunii peste tot incercand sa gaseasca butonul de Ok, de exemplu. Un astfel de soft nu poate decat sa ajunga o relicva de care isi va aduce aminte doar Wikipedia. Pana la urma, cine ar vrea sa foloseasca un soft in care nu poate sa efectueze nici cele mai de baza operatii?[1]

A doua regula a proiectarii unei interfete reusite se refera la folosirea asa-numitelor „metafore” si „affordances”. O metafora reprezinta un mod de a crea, in mintea utilizatorului, o conexiune intre o actiune realizata in software si un obiect/actiune din lumea reala, pe care utilizatorul este mult mai probabil sa o recunoasca.[1]

Un exemplu foarte bun de metafora este butonul de Zoom, care se gaseste in aproape orice software. Daca butonul are si o iconita atasata, acea iconita va fi intotdeauna o lupa. Astfel, utilizatorul care doreste sa mareasca o anumita portiune dintr-o fotografie, de exemplu, nu va avea probleme in a folosi butonasul cu lupa, fara sa ii fie frica de modificarea dimensiunilor imaginii, deoarece stie ca lupa nu face asta in realitate. Un exemplu de „affordance” este coltul dreapta-jos al ferestrelor de Windows, care arata ca si cum ar fi facute special pentru aderenta, sugerand utilizatorului ca daca trage cu mouse-ul de el va redimensiona fereastra.[1]

Probabil cea mai cunoscuta metafora, desi putini realizeaza asta, este chiar Desktop-ul din Windows. Organizat cu fisiere si dosare, desktop-ul „simuleaza” biroul unei persoane. Iar utilizatorilor li s-a parut cat se poate de normal sa mute fisierele si sa le organizeze in foldere, deoarece asta faceau si in realitate pe birourile lor.[1]

Cu toate aceastea, o metafora prost aleasa poate crea atat confuzie, cat si frustrare. Ce s-ar intampla daca butonul de Zoom mentionat mai devreme ar mari efectiv dimensiunea imaginii? Sau daca butonul de selectare dintr-un program de editare de imagini ar actiona pe post de buton de decupare? Ne alegem cu un utilizator confuz care, cel mai probabil, nu isi va da seama ce a facut, si, mai ales, cum sa repare greseala(in cazul de fata un simplu Ctrl-Z este de ajuns, insa in alte cazuri aceasta optiune poate sa nu fie disponibila). Sansele sunt ca utilizatorul sa paraseasca exasperat aplicatia, si sa nu o mai foloseasca niciodata, pe motiv ca „nu merge, are buguri”.[1]

Ceea ce ne aduce la ultimul aspect al discutiei, aspect care e strans legat de primele doua: interfata unui software trebuie sa faca exact ce se asteapta utilizatorul sa faca. In exemplul de mai sus, in cele doua situatii nu se poate vorbi de bug-uri, ci de functionalitati pe care le are orice program decent de editare de imagini(mai putin Paint), doar ca metaforele alese pentru acele functionalitati erau total gresite. Daca metaforele ar fi fost alese astfel incat sa faca ceea ce utilizatorul se asteapta sa faca, tanti Lili, care are 80 de ani si nu isi recunoaste nepotii in poze decat daca fata lor apare pe jumatate de ecran, ar fi fost mai fericita. La fel si Mihai, care ar fi gasit butonul de Crop prin care si-ar fi putut elimina sotia din pozele de vacanta pe care i le va trimite secretarei. Un alt exemplu(din realitate) este comportamentul ferestrelor pe Windows vs. Mac. Pe Windows, clickul pe coltul din dreapta jos al ferestrei va redimensiona fereastra, in timp ce pe Mac acelasi click va muta fereasta. Acesta este un exemplu de comportament diferit, care, desi nu este gresit, va parea neintuitiv unui utilizator ce a fost obisnuit cu un sistem de operare si incearca sa treaca la unul nou.[1]

Cand vine vorba de proiectarea unei interfete, este foarte bine sa ne imaginam ce fel de utilizatori o vor folosi, si sa incercam sa empatizam cu ei. Da, poate eu stiu sa scriu ls –la | grep fisier.doc pentru a gasi un fisier intr-un folder suprapopulat, dar sunt foarte sigur ca secretara sefului nu stie. Ea prefera casuta de search. Treaba interfetei e sa simplifice lucrurile fata de vechea linie de comanda, si acest lucru este util chiar si utilizatorilor avansati.

Acest proiect propune dezvoltarea unor interfete pentru anumite baze de date in care un utilizator poate usor accesa continutul acesteia.[1]

* 1. MySQL

MySQL este un sistem de gestiune a bazelor de date relaționale, produs de compania suedeza MySQL AB și distribuit sub Licența Publică Generală GNU. Este cel mai popular SGBD open-source la ora actuală, fiind o componentă cheie a stivei LAMP (Linux, Apache, MySQL, PHP).[3][10][11]

Deși este folosit foarte des împreună cu limbajul de programare PHP, cu MySQL se pot construi aplicații în orice limbaj major. Există multe scheme API disponibile pentru MySQL ce permit scrierea aplicațiilor în numeroase limbaje de programare pentru accesarea bazelor de date MySQL, cum ar fi: C, C++, C#, Java, Perl, PHP, Python, FreeBasic, etc., fiecare dintre acestea folosind un tip specific API. O interfață de tip ODBC denumită MyODBC permite altor limbaje de programare ce folosesc această interfață, să interacționeze cu bazele de date MySQL cum ar fi ASP sau Visual Basic. În sprijinul acestor limbaje de programare, unele companii produc componente de tip COM/COM+ sau .NET (pentru Windows) prin intermediul cărora respectivele limbaje să poată folosi acest SGBD mult mai ușor decât prin intermediul sistemului ODBC. Aceste componente pot fi gratuite (ca de exemplu MyVBQL) sau comerciale. LAMP cuprinde și MySQL Licența GNU GPL nu permite încorporarea MySQL în softuri comerciale; cei care doresc să facă acest lucru pot achiziționa, contra cost, o licență comercială de la compania producătoare, MySQL AB .[3][12][14][8]

MySQL este componentă integrată a platformelor LAMP sau WAMP (Linux/Windows-Apache-MySQL-PHP/Perl/Python). Popularitatea sa ca aplicație web este strâns legată de cea a PHP-ului care este adesea combinat cu MySQL și denumit Duo-ul Dinamic. În multe cărți de specialitate este precizat faptul ca MySQL este mult mai ușor de invățat și folosit decât multe din aplicațiile de gestiune a bazelor de date, ca exemplu comanda de ieșire fiind una simplă și evidentă: „exit” sau „quit”. Pentru a administra bazele de date MySQL se poate folosi modul linie de comandă sau, prin descărcare de pe internet, o interfață grafică: MySQL Administrator și MySQL Query Browser. Un alt instrument de management al acestor baze de date este aplicația gratuită, scrisă în PHP, phpMyAdmin.[3][14]

MySQL poate fi rulat pe multe dintre platformele software existente: AIX, FreeBSD, GNU/Linux, Mac OS X, NetBSD, Solaris, SunOS, Windows 9x/NT/2000/XP/Vista.[3]

In Figura 1 sunt prezentate tabelele create in baza de date. Am avut de realizat o baza de date ce contine tabele propuse la care am avut de adaugat o tabela de lagatura de tipul M:N.[4]

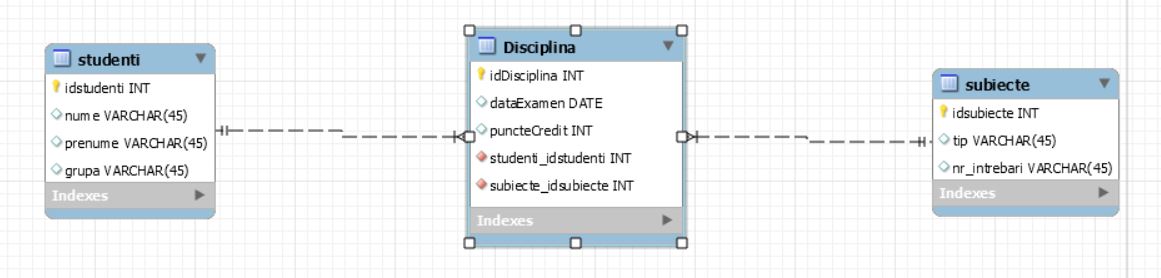


Figura 1.1 – Diagrama proiectului

Ce sunt asocierile de tip many-to-any?

Aceste tipuri de asocieri apar în prima fază a proiectării bazei de date, însă ele trebuie să fie ulterior eliminate. Figura 1.2 prezintă câteva exemple de relaţii many-to-many. La punctul b am considerat că un curs poate apărea pe oferta de cursuri a unei facultăţi, însă poate să nu fie aleasă de nici un student de aceea un curs poate fi urmat de unul sau mai mulţi studenţi. Invers, este posibil ca un student să fi terminat studiile şi să se pregătească pentru susţinerea examenului de licenţă şi de aceea el nu mai frecventează nici un curs. La punctul c, un profesor angajat al unei şcoli trebuie să predea cel puţin o disciplină. Iar o disciplină din planul de învăţământ trebuie să fie predată de cel puţin un profesor .[3][2][4]

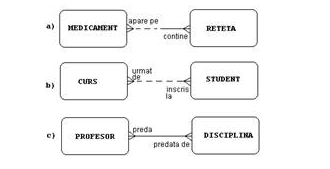


Figura 1.2 – Asocieri many-to-many

CAPITOLUL 2

JAVA SERVER PAGES

* 1. Descriere

Tehnologia Java Server Pages (JSP) este cea mai populara metoda de a crea interfete Web pentru aplicatiile care ruleaza pe platforma Java, creata de Sun. Ea se bazeaza pe tehnologia numita Java Servlets fiind, de fapt, o completare a acesteia in ideea crearii cat mai facile a paginilor Web dinamice.[2]

Punctul central al tehnologiei o reprezinta asa-numitele pagini JSP care sunt, practic, fisiere text care combina descrieri HTML cu cod Java. Paginile JSP sunt gestionate si accesibile prin intermediul unui server de aplicatii. Acesta primeste cereri venite prin HTTP de la un browser Web. Daca o cerere refera o pagina JSP, serverul prelucreaza local pagina respectiva si, in functie de continutul acesteia, genereaza dinamic o pagina HTML pe care o trimite, ca raspuns, browser-ului. Este important de retinut faptul ca toate prelucrarile legate de paginile JSP se fac pe partea de server, acestea nefiind niciodata transmise in forma originala catre client. In plus, trebuie retinut faptul ca serverul de aplicatii include si o masina virtuala Java in care ruleaza atat codul Java intalnit in paginile JSP cat si obiectele instantiate de acesta. Pentru cei interesati, facem precizarea ca procesul de prelucrare pe partea de server a paginilor JSP presupune, de fapt, crearea unor clase Java Servlet care urmeaza regulile scrise in pagina JSP si include codul Java din aceasta. Clasele astfel generate sunt apoi compilate si rulate in masina virtuala amintita.[2][6][16][17]

Un alt element important este ca orice aplicatie Web JSP trebuie sa fie instalata (deployed) in serverul de aplicatii inainte de a putea fi rulata. Instalarea presupune copierea paginilor JSP si a claselor Java folosite de acestea in locatii bine stabilite de catre serverul de aplicatie utilizat, eventual intr-o forma arhivata de tip .jar (Java Archive) sau .war (Web Archive). [2][19][20]

* 1. Modul de functionare

Pe măsură ce Web s-a dezvolat şi a început să fie utilizat pentru oferirea de servicii, a apărut necesitatea de a construi pagini cu conţinut dinamic. Applet-urile reprezintă un exemplu de tehnologie, ce utilizează platforma clientului pentru a oferi un conţinut dinamic.[5][16]

La nivelul serverelor de web, printre primele tehnologii folosite pentru creare de conţinut dinamic a fost Common Gateway Interface (CGI). Ulterior au apărut şi s-au dezvoltat tehnologii similare (PHP, ASP) care oferă metode pentru construirea de pagini ce permit interacţiunea cu utilizatorul. Tehnologia propusă de firma SUN pentru construirea de pagini cu conţinut dinamic este tehnologia Servlet. Această tehnologie este bazată pe limbajul Java, componentele de timp servlet fiind implementate în acest limbaj. Bazat pe tehnologia Servlet s-au dezvoltat ulterior tehnologii ca Java Server Pages (JSP) şi Java Server Faces (JSF) -ambele destinate construirii de pagini cu conţinut dinamic.[17]

Un servlet reprezintă o componentă web , gestionată de un container , care generează conţinut dinamic. Servlet-urile sunt clase Java, ce oferă independenţă de platformă şi sunt incărcate şi executate dinamic de către server. Servlet-urile comunică cu clienţii pe baza paradigmei cerere – raspuns. Acest model cerere – raspuns se bazează de obicei pe protocolul Hypertext Transport Protocol (HTTP).[16]

Containerul de servlet-uri este o componentă ce oferă servicii de reţea prin intermediul cărora servleturile primesc şi transmit cereri şi răspunsuri de la şi către clienţi. Containerul de servlet-uri înmagazinează servleturile şi este responsabil pentru gestionarea acestora. Un container poate exista în cadrul unui server web sau poate fi adăugat ulterior utilizând mecanismul de extensie al serverului. Modelul de acces e prezentat mai jos.[18]

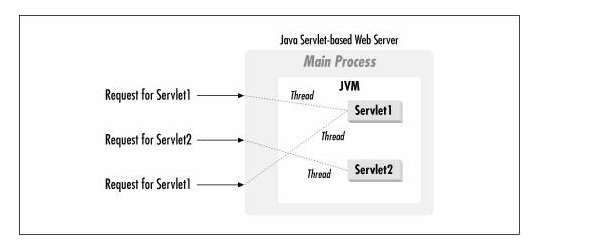


Figura 2.1 - Servlet

Exemplu: Un program client (web browser) accesează un server web şi transmite o cerere HTTP (poate fi de exemplu un form completat cu datele unei persoane). Acceastă cerere este preluată de către serverul de web şi în cazul în care este destinată unui servlet, este transmisă mai departe către containerul de servleturi. Containerul determină cărui servlet îi este adresată cererea şi va invoca respectivul servlet, transmiţînd-ui ca parametri două obiecte cerere (request) şi raspuns (response).[1]

Servlet-ul va utiliza obiectul request pentru a determina cererea făcută de clientul web. După realizarea operaţiilor necesare (de exemplu scrierea sau citirea unor date dintr-o bază de date ), servletul va tansmite către client un raspuns prin intermediul obiectului response.[1]

In continuare este prezentată structura de bază a unui servlet Fi

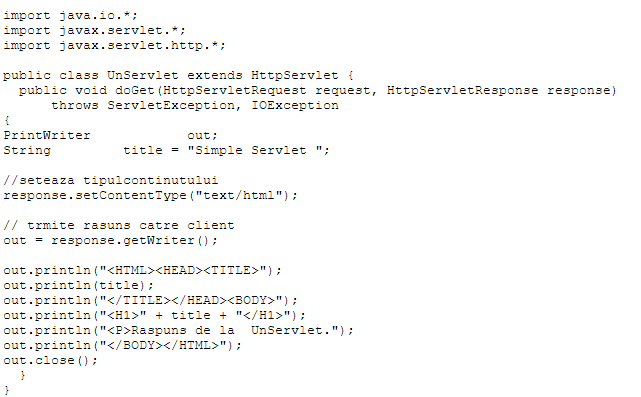


Figura 2.2 – Structura de bază a unui servlet Fi

Pentru a construi un servlet care lucrează prin protocolul HTTP trebuie extinsă clasa HttpServlet. In continuare este suprascrisă metoda doGet(…) pentru a prelucara cereri GET. De asemenea poate fi rescrisă şi metoda doPost pentru a prelucra cereri POST.[1]

Metodele doGet() şi doPost() primesc ca argumente doua obiecte: HttpServletRequest şi HttpServletRespons. Obiectul HttpServletRequest este utilizat pentru a determina cererea pe care a făcut-o clientul (datele din FORM , header HTTP etc.). Pentru aceasta clasa pune la dispoziţie metode cum ar fi: getHeaders(), getMethod(), getPathInfo(), getParameter(), getInputStream(), getReader().[1]

Obiectul HttpServletResposn este utilizat pentru a transmite raspunsul către client. Acesta conţine metodele necesare pentru a stabili headerul, tipul de raspuns şi pentru a obţine fluxul de ieşire prin intermediul căruia răspunsul este transmis către client. Clasa pune la dispoziţie metode cum ar fi: setHeader(), setStatus(), getWriter, getOutputStream() etc.

Observaţie: Metodele doGet() şi doPost() sunt apelate de către metoda service(), şi există posibilitatea să se rescrie metoda service(), prin intermediul acesteia fiind prelucrate atât cererile POST cât şi cererile GET.[1]

Diferenta dintre metodele doGet() si doPost() sunt prezente in tabelul din figura 2.3 si Figura 2.4:

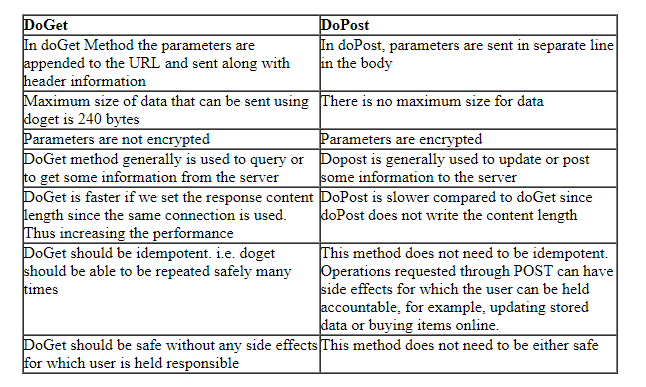


Figura 2.3 - Tabel ce prezintă metodele DoGet și DoPost

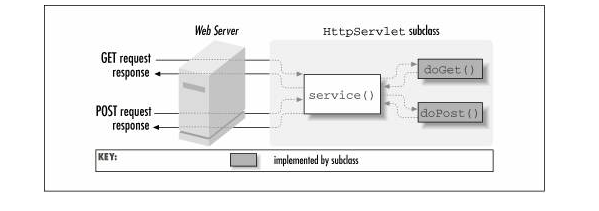


Figura 2.4 - Imagine ce prezintă metodele DoGet și DoPost

Ciclul de viata a unui servlet cuprinde următoarele etape:

1. creare şi iniţializare servlet. In momentul în care un servlet este încărcat în memorie va fi apelată metoda init(ServletConfig cfg)

2. preluare si deservire cereri sosite de la clienti

3. distrugere servlet si eliberare memore. In momentul în care un servlet este distrus va fi apelată metoda destroy()

In momentul în care un servlet este invocat pentru prima dată containerul de servleturi va încărca o instanţă a acestui servlet în memorie. Această instanţă va prelua şi deservi toate cererile ce sosesc de la clienţi pentru acest servlet. Existenţa unei singure instanţe a unui servlet în memorie, ce deserveste toţi clienţii prezintă avantaje prin faptul că:

* reduce memoria ocupată de servlet-uri
* micşorează timpul de răspuns şi resursele de procesor cerute pentru deservirea unui client de către un servlet, deoarece nu trebuie construită câte o instanţă a servlet-ului pentru fiecare cerere a unui client
* La iniţializare, un servlet poate iniţializa toate resursele de care are nevoie, acestea persistând pe parcursul apelurilor de la diverşi clienţi. Un exemplu este iniţializarea unei conexiuni la baza de date ce mai apoi poate fi reutilizată pentru a deservi toţi clienţii

Servlet-ul va fi încărcat în memorie la prima invocare a acestuia de către un client, după care această instanţă creată va deservi toţi clienţii. Din punctul de vedere al programatorului unui servlet, fiecare client poate fi văzut ca un nou thread ce apelează metodele service(), doGet() sau doPost().[1]

Marea majoritate a containerelor de servlet-uri permit comunicarea între servlet-urile încărcate în memorie, astfel încât o cerere transmisă către un servlet poate fi redirectionata către un alt servlet, pagina de răspuns fiind transmisă către client de ultimul servlet din lanţ ca in Figura 2.4.

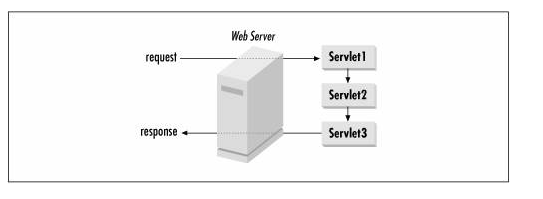


Figura 2.5

CAPITOLUL 3

PREZENTAREA CODULUI

Fisierul care sta la baza intregului program si in care sunt construite functiile amintite mai sus este JavaBean.java.

Mai jos, in Figura 3.1 e prezentat una din functiile prezente, mai exact adaugaSubiecte, ce are ca scop adăugarea unei linii in tabela “subiecte” din baza de date MySQL.[4]

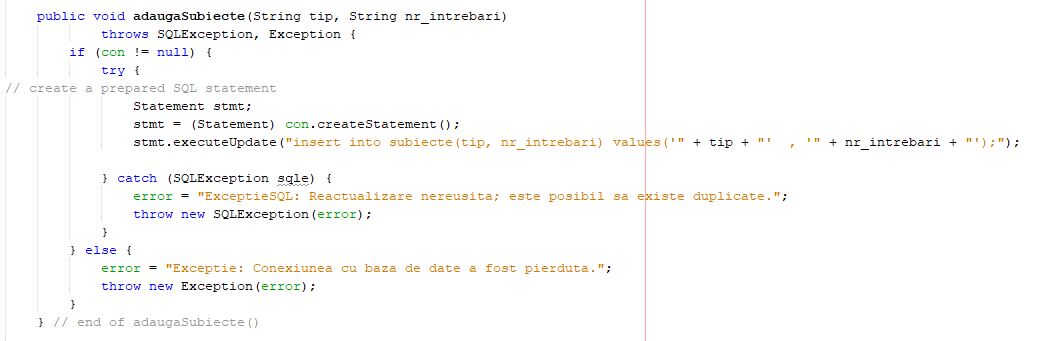


Figura 3.1 – Funcția de adăugare subiecte

Similar au fost implementate si functiile adaugaStudenti si adaugaDisciplina, ce au ca scop adaugarea unei linii in tabela ”studenti”, respectiv in tabela “Disciplina” din baza de date MySQL.

De asemenea, mai departe urmeaza sa fie prezentate si functiile de:

* Vizualizare tabelă

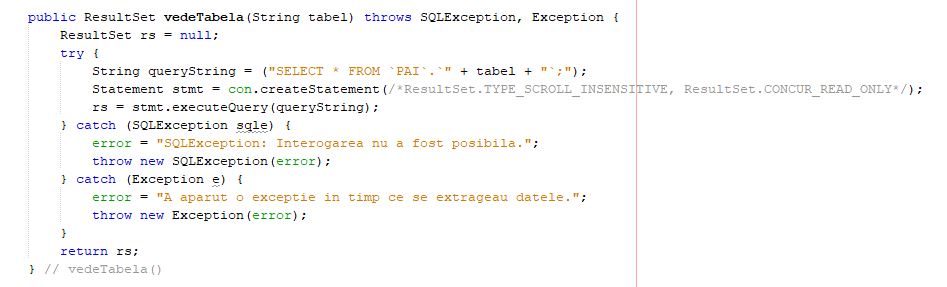


Figura 3.2 – Funcția de vizualizare tabelă

* Stergere tabelă

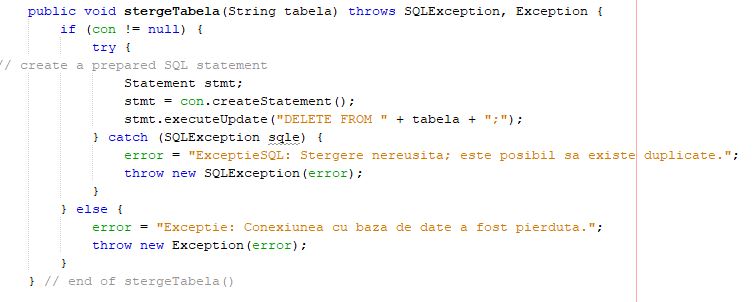


Figura 3.3 – Funcția de ștergere tabelă

* Modifică tabelă

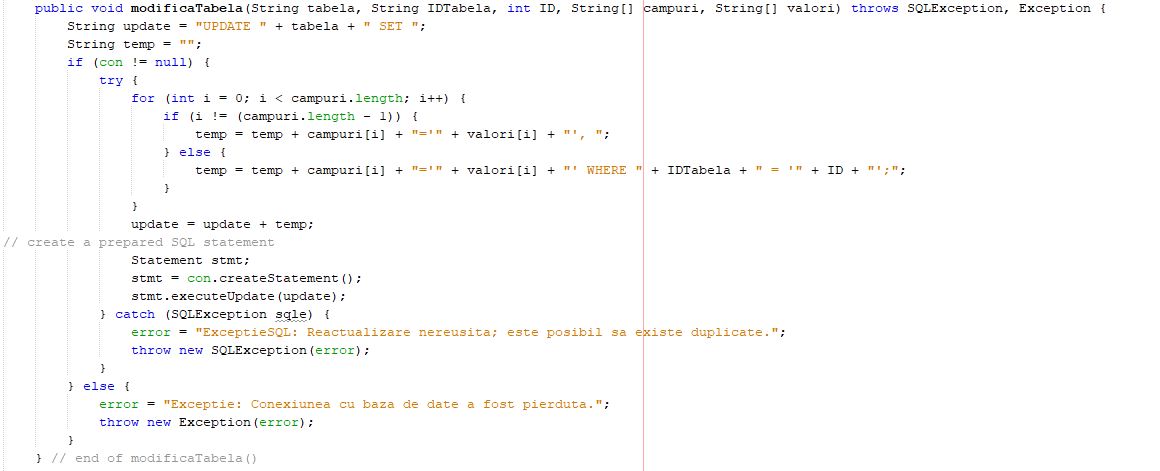


Figura 3.4 – Funcția de modificare tabelă

Mai jos, în Figura 3.5, am ales să prezint și funcția care face apelul către server.

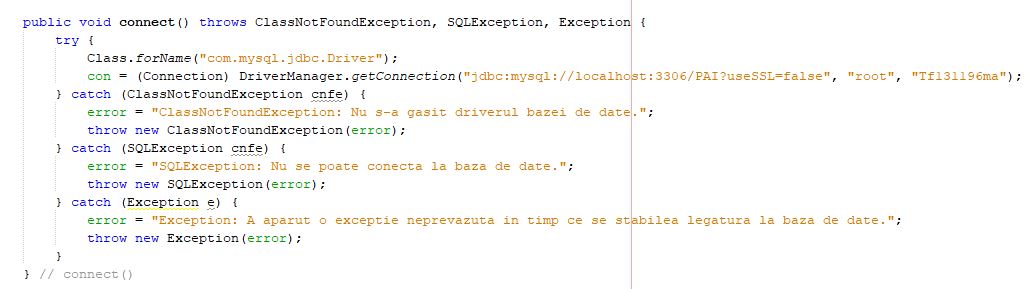


Figura 3.5 – Funcția de conectare la server

JSP-urile au acces la intreaga familie Java API (Application Programming Interface), incluzand si JDBC (Java Database Connectivity), cea din urma definind modul in care un utilizator poate accease baza de date. O component JSP este create pentru a prelua rolul de interfata cu utilizatorul pentru o aplicatie Java Web. Dezvoltatorii Web combina in fisierele JSP cod HTML, elemente de tip XML, precum si actiuni si comenzi JSP incorporate. Folosind JSP se pot colecta date introduce de utilizatori prin formulare Web, se pot prezenta date dintr-o baza de date (sau dintr-o alta sursa), si se pot crea pagini Web dinamice.[16]

În continuare am să prezint și modul în care o sa arate HOME page-ul, figura 3.6 si codul in figura 3.7 din aplicatia mea, si codul din fisierul .html care sta la baza acesteia.[4]

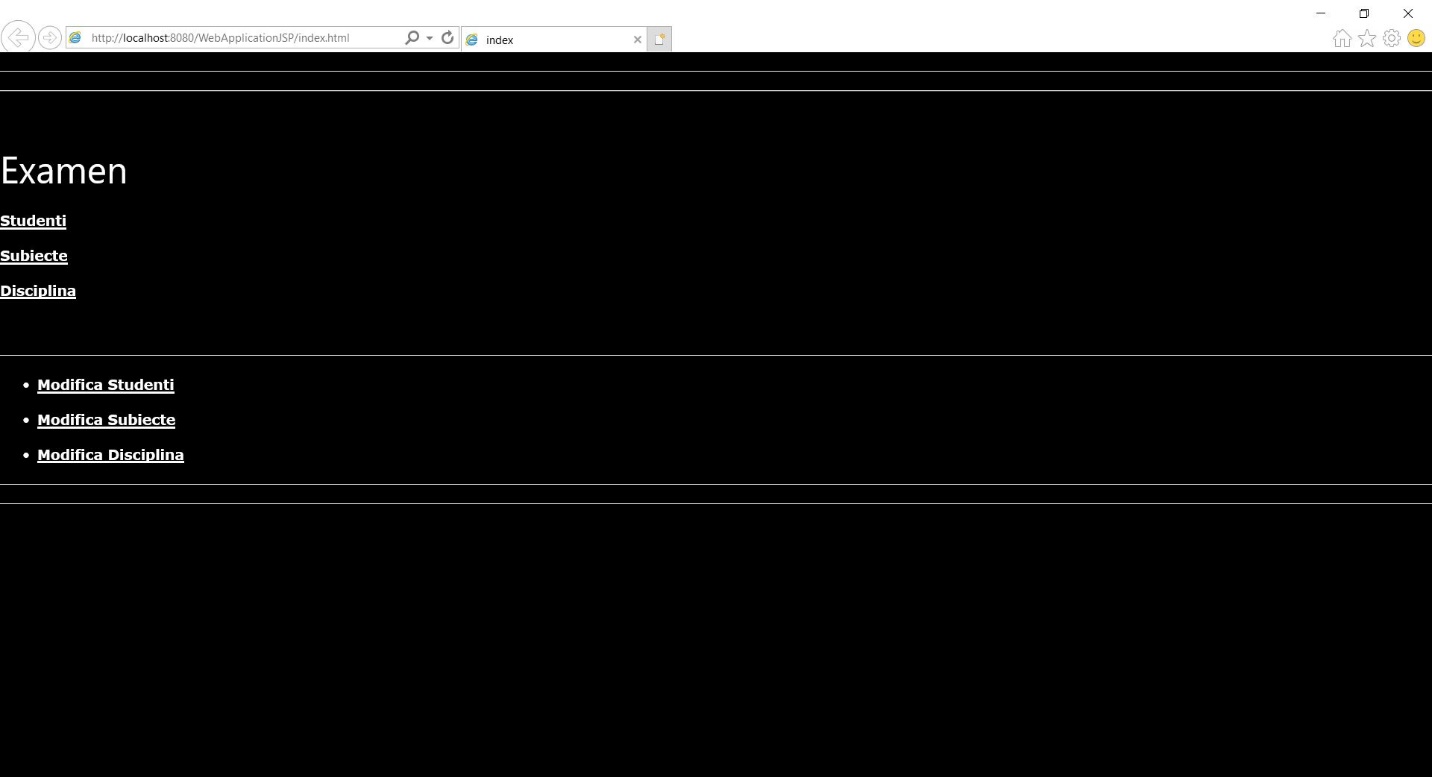


Figura 3.6 – Pagina principală



Figura 3.7 – Codul din spatele paginii principale

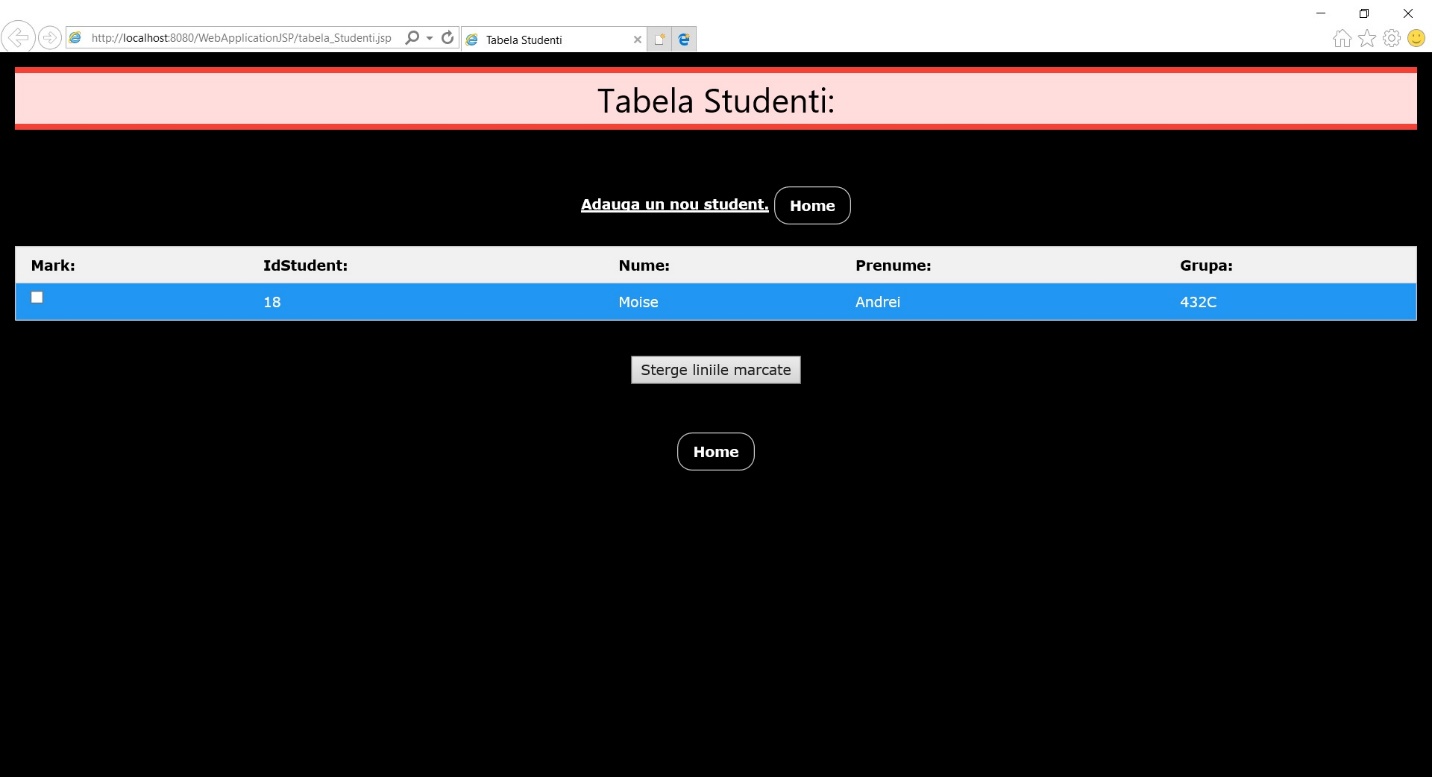


Figura 3.8 – Tabela “studenti”

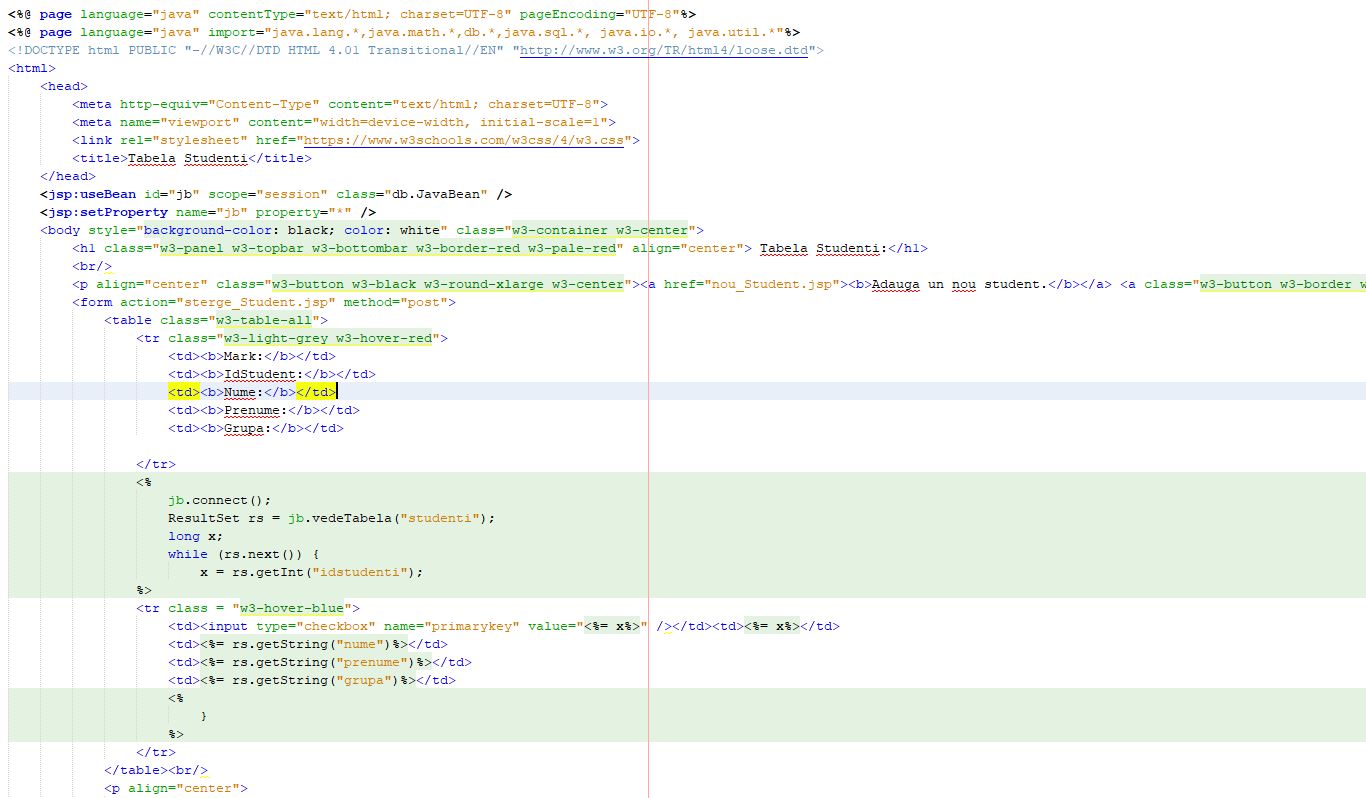




Figura 3.9 – Codul asociat tabelei “studenti”

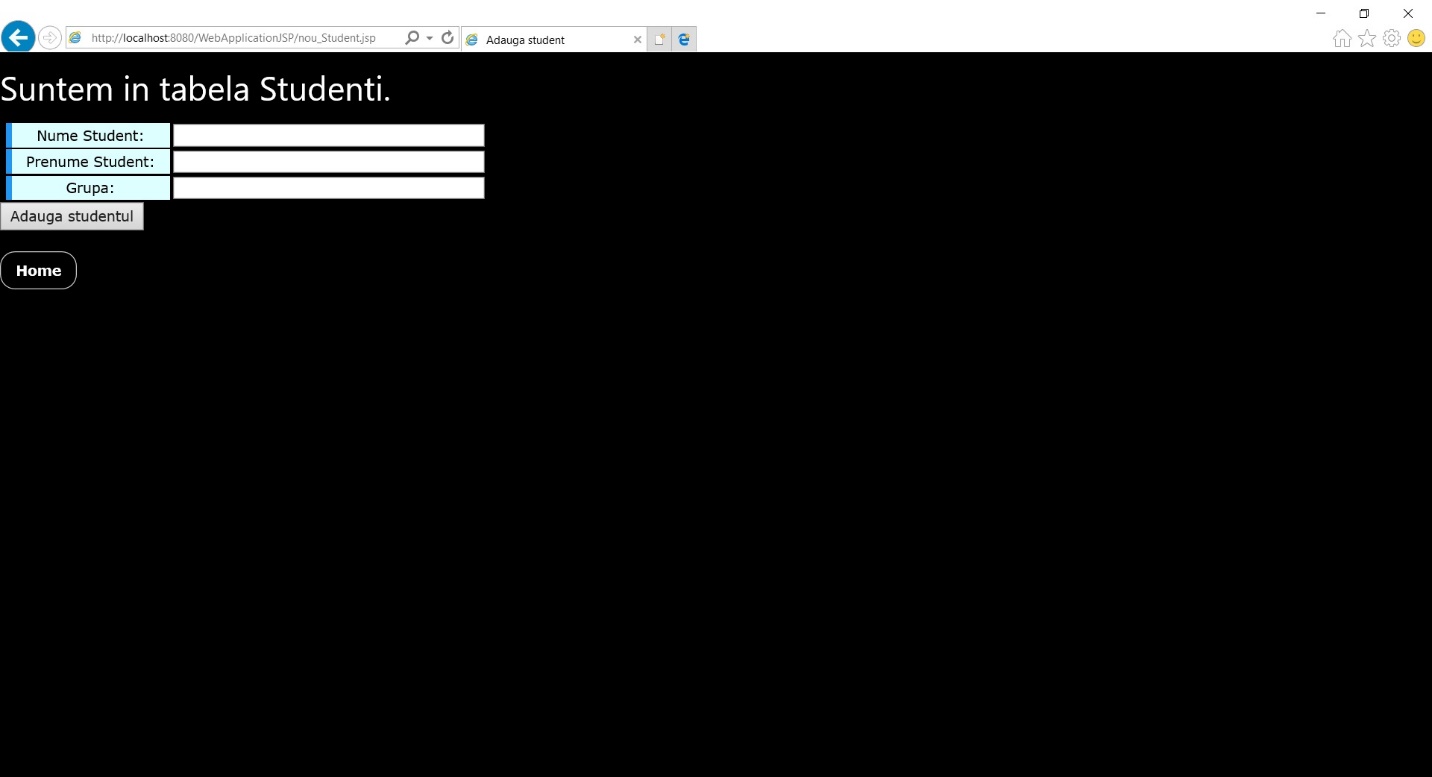
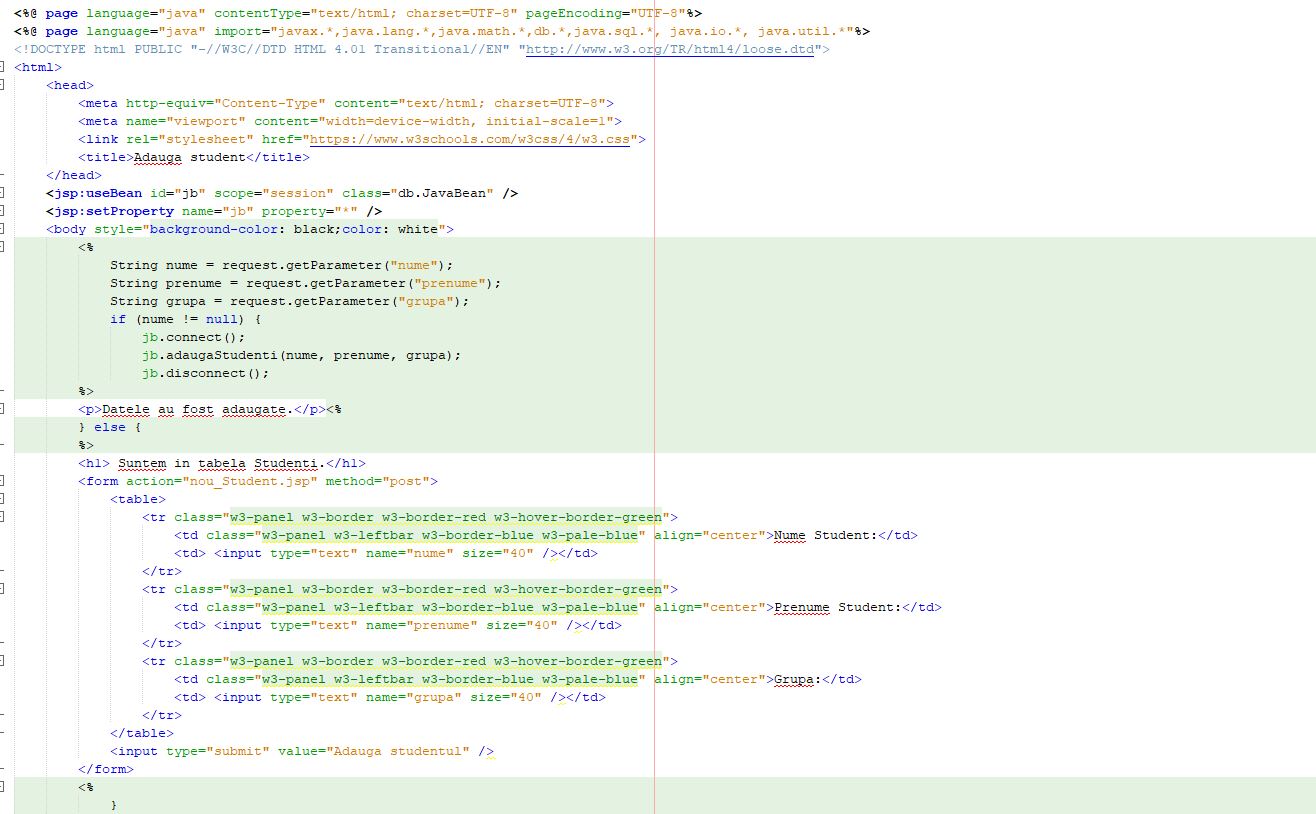


Figura 3.10 – Functia de adaugare student



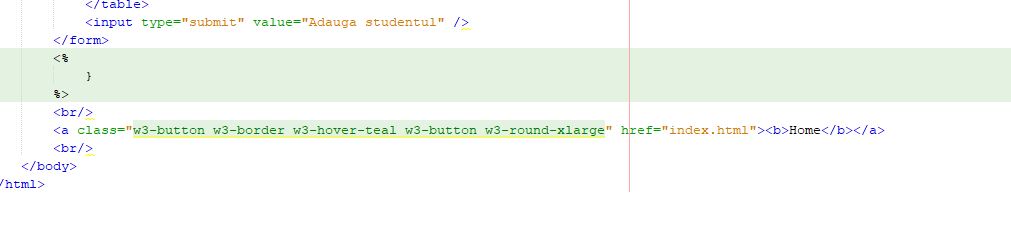


Figura 3.11 – Codul asociat functiei de adaugare student

Bibliografie

1. http://www.ligasoftware.ro/blog/2011/03/despre-proiectarea-interfetelor
2. http://www.atestateinformatica.ro/tutoriale/relatii-intre-entitati
3. 3.
4. [https://ro.wikipedia.org/wiki/MySQL](https://ro.wikipedia.org/wiki/MySQL )
5. Curs Programare aplicativa a interfetelor, Valentin Pupezescu
6. Dan C. Cosma, "Programarea aplicatiilor distribuite", Editura de Vest, 2009, Timisoara, ISBN 978-973-36-0501-0.
7. <http://control.aut.utcluj.ro/scd/lab6/tehnologiiWeb.htm>
8. <https://pingendo.com/>
9. <https://www.mysql.com/products/workbench/>
10. <https://netbeans.org/>
11. Changes in MySQL 5.6.19”. MySQL 5.6 Reference Manual. Oracle. 30 mai 2014. Accesat la 30 mai 2014.
12. „Changes in Release 5.7.4 (Development)”. MySQL 5.7 Reference Manual. Oracle Corporation. 31 martie 2014. Accesat la 2 aprilie 2014.
13. „MySQL: Project Summary”. Ohloh. Black Duck Software. Accesat la 17 septembrie 2012.
14. „Supported Platforms: MySQL Database”. Oracle. Accesat la 24 martie 2014.
15. MySQL :: MySQL Downloads
16. DB-Engines Ranking
17. [Mailing list archive: "Sun JSP 1.0 \*not\* available"](http://www.xent.com/FoRK-archive/apr99/0822.html)
18. [The Life Cycle of a JSP Page (Sun documentation)](http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/JSPIntro4.html)  [Understanding JavaServer Pages Model 2 architecture (JavaWorld)](http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.html)
19. [Forum thread (JavaRanch): OutputStream already obtained](http://www.coderanch.com/t/286297/JSP/java/OutputStream-already-obtained#2107162)
20. [JSTLs documentation (Oracle)](http://docs.oracle.com/javaee/5/tutorial/doc/bnakc.html) bb
21. [Tag Libraries Tutorial - What is a Tag Library? (Sun)](http://java.sun.com/products/jsp/tutorial/TagLibraries3.html#63159)