Cuprins

[1. Descrierea domeniului economic 1](#_Toc389231109)

[1.1. Definirea problemei 1](#_Toc389231110)

[1.2. Modul actual de gestionare al evenimentelor 1](#_Toc389231111)

[1.3. Necesitate 3](#_Toc389231112)

[1.4. Funcționalități noi ale sistemului prezentat 3](#_Toc389231113)

[2. Tehnologii informatice folosite 4](#_Toc389231114)

[2.1. Limbaje și platforme folosite 4](#_Toc389231115)

[2.1.1. Java 7 4](#_Toc389231116)

[2.1.2. Spring Framework 5](#_Toc389231117)

[2.1.3. SQL 12](#_Toc389231118)

[2.2. Programe folosite 12](#_Toc389231119)

[2.2.1. SpringSource Tool Suite 12](#_Toc389231120)

[2.2.2. Apache Tomcat 7.0 13](#_Toc389231121)

[3. Proiectarea aplicației 14](#_Toc389231122)

[3.1. Proiectarea bazei de date 14](#_Toc389231123)

[3.2. Diagrame UML 15](#_Toc389231124)

[4. Prezentarea soluției 20](#_Toc389231125)

[5. Stabilirea necesarului de resurse 26](#_Toc389231126)

[Bibliografie 27](#_Toc389231127)

[Anexa A 28](#_Toc389231128)

# 2. Tehnologii informatice folosite

## 2.1. Limbaje și platforme folosite

### 2.1.1. Java

Java este un limbaj de programare orientat pe obiecte, independent de platformă, portabil, sigur, robust , care poate rula pe mai multe fire de execuție. Java este atât un limbaj de programare cât și o platforma. Spun că este o platforma, deoarece are propriul mediu de execuție (JRE).

Java este folosit în cam tot ce se dezvolta nou, asa că aplicațiile desktop (Acrobat Reader, media player, antivirus etc.), aplicațiile web, aplicațiile mobile, jocurile și multe altele profită de avantajele acestui limbaj.

Cele mai importante tipuri de aplicații Java sunt:

* *aplicațiile desktop*, sunt acele aplicații pe care le instalam unde avem nevoie (exemplu: media player, antivirus). Astfel de aplicații sunt dezvoltate în AWT și SWING;
* *aplicațiile web,* sunt aplicațiile care rulează pe partea de server și creează pagini dinamice. Astfel de aplicații sunt create utilizând servleti, JSP-uri, Struts, JSF, etc;
* *aplicațiile enterprise ,* sunt aplicațiile mai complexe, cu o securitate mai mare (exemplu: aplicație bancară) . Astfel de aplicații sunt create folosind EJB-uri;
* *aplicațiile mobile,* sunt toate aplicațiile create pentru mobil. Astfel de aplicații sunt create folosind Android.

Java este folosit în special pentru că aplicațiile dezvoltate sunt independente de platfomă. Acest

lucru se întâmplă deoarece codul scris este compilat și transformat în bytecode, se obțin fișiere cu extensia .class, iar aceste fișiere pot fi rulate pe orice platformă. Acest lucru este ilustrat în figura de mai jos.

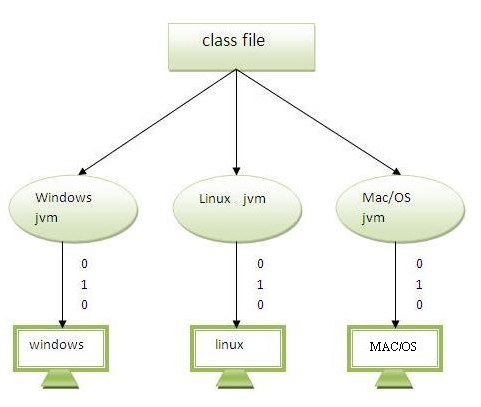


Figura 1. Fișier de tip class

### 2.1.2. Maven

Maven este un framework care are ca și scop ușurare procesului de build pe un proiect, și rezolvarea dependințelor. Înainte să apară acest proiect lucrurile stăteau în felul următor:

* librăriile erau adăugate în fiecare proiect;
* proiectul trebuia să aibă o anumită structură , altfel nu putea fi pornit;
* build-ul și deploy-ul durau mai mult.

Acum cu maven, viața dezvoltatorilor este mai simplă, totul este configurabil și lucrurile sunt mai simple. În continuare voi prezenta pe pași cum am procedat pentru pornirea proiectului cu maven:

* am luat o versiune de maven recentă (Apache Maven 3.1.1) și am dezinstalat-o pe local;
* am adăugat în variabilele de sistem variabila MAVEN\_HOME, unde am setat calea către maven;
* am verificat că totul este configurat rulând comanda: ”mvn –version”;
* am creat un poiect nou în care am adăugat un fișier pom.xml, în care voi trece toate librările de care am nevoie în proiect;

În fișierul pom.xml sunt defapt multe taguri cu anumite roluri. Voi spune câteva cuvinte despre cele mai importante:

......

**Spring**

Spring este un framework care oferă suport pentru dezvoltarea aplicațiilor web. Cum în lucrarea de față folosesc acest framework, voi prezenta pe scurt cele mai importante caracteristici ale acestuia.

Spring vine cu o grămadă de avantaje, mai jos sunt cele mai importante:

* oferă template-uri pentru JDBC, Hibernate, JPA și alte tehnologii, în felul acesta se scrie mai puțin cod, deoarce ascunde câțiva pași de bază. Un exemplu concret ar fi JdbcTemplate, unde nu va mai fi nevoie să ma ocup de crearea conexiunii, crearea statement-ului, închiderea conexiunii, tot ce trebuie să fac este să scriu query-ul;
* aplicațiile scrise cu Sping sunt independente datorita principiului de dependency injection, pe care îl voi prezenta mai pe larg în cele ce urmează;
* acest framework face ca aplicațiile să fie ușor de testat;

În cadrul acestui framework găsesc mai multe module: beans, AOP, JDBC, ORM, OXM, JMS, tranzacții, servleti, etc. Aceste module sunt grupate ca în figura de mai jos:

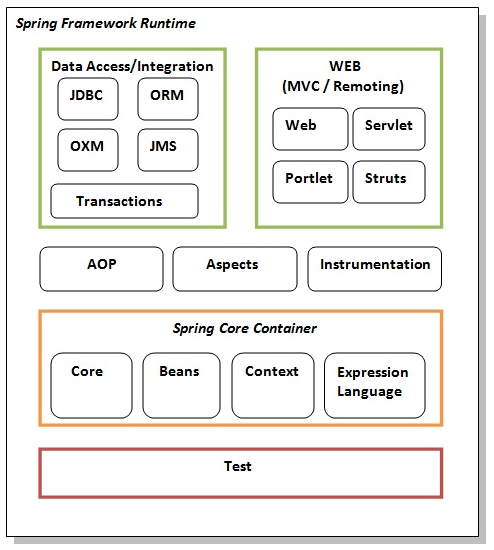


Figura 2. Modulele din Spring

**Dependency Injection (DI)**

Această tehnologia este element cheie în acest framework. Să presupunem că scriem o aplicație Java complexă, unde clasele trebuie să fie independente între ele, pentru a crește posibilitatea de reutilizare a acestora. Dependency Injection ajută aceste clase să fie independente, dar oferă și posibilitatea de a le lega la un moment dat. Pare ciudat la o primă vedere, dar să luăm cuvintele separat. Cuvântul ”dependency” s-ar putea traduce ca o legătură între doua clase (de exemplu clasa A depinde de clasa B). Cuvantul ”injection” reflectă modul în care se va face această legătură, mai exact clasa B va fi inclusă în clasa A. Acest lucru se întâmplă cu ajutorul a ceea ce se numește Inversion of Control (IoC).

Pentru a înțelege mai bine conceputul voi lua un exemplu concret. Să zicem că avem clasele ”Student”, ”Nota” și vrem să legăm clasa ”Student” de clasa ”Nota”.

În exemplul de mai jos avem un exemplu de legătură cu și fără dependency injection:

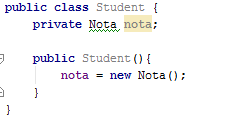
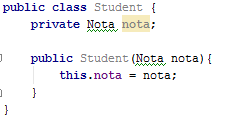
 

Figura 3. Exemplu fără DI Figura 4. Exemplu cu DI

În cel de-al doilea exemplu, clasa ”Student” nu trebuie să știe implementările din clasa ”Nota”. Clasa ”Nota” va fi implementată separat și va fi furnizată clasei ”Student”, în momentul în care aceasta este inițializată. Astfel păstrăm cele doua clase independente una de cealaltă, dar legătura este creată prin intermediul constructorului.

**IOC Container**

Container-ul Spring este cel care creează obiectele, le leagă între ele, le configurează și se ocupă de tot ciclul de viață al acestora. Acest container folosește DI pentru a lega obiectele între ele. De fapt container-ul primește instrucțiuni legate de ce obiecte trebuie să inițializeze prin citirea unui fișier de configurare. Acest fișiser de configurare poate fi reprezentat printr-un fișier XML, adnotări Java sau cod Java. În figura de mai jos este prezentat cum funcționează container-ul spring. Practic acesta se folosește de clase Java numite POJO (Plain Old Java Object) si fișiere de configurare pentru a produce o aplicație stabilă.

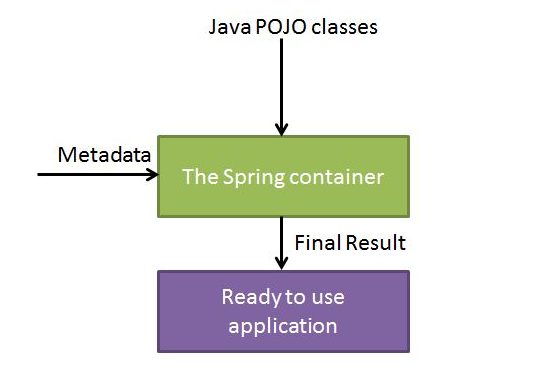


Figura 5. Conainerul Spring

**Spring MVC**

Spring MVC este un framework care se bazează pe o arhitectură de tipul model-view-controller, care are ca și scop crearea unei aplicații web ușor de dezvoltat, menținut și testat. Principiul de bază al acestui framework este separarea anumitor aspecte ale unei aplicații web. Elementele de tip ”model” vor încapsula partea de date dintr-o aplicație. Elementele de tip ”view” se vor ocupa de modul în care sunt afișate datele la client. Elementele de tip ”controller” vor procesa cererile de la client si vor returna modelul corespunzător, care ulterior va fi prelucrat de ”view”.

Acest framework folosește DispacherServlet pentru gestionarea tuturor cererilor venite de la client. Practic un request HTTP venit de la client ajunge în DispacherServlet. De aici în funcție de tipul de request, cererea este asignată unui controller. Controller-ul este cel care primește requestul și în funcție de tipul de request apelează o metoda GET sau POST. Metoda respectivă va seta datele pe model și va returna numele view-ul corespunzător. DispacherServlet se folosește de ViewResolver pentru a returna în funcție de numele primit din controller pagina (view-ul) corespunzător. Într-un final view-ul este returnat în browser. Mai jos este prezentat grafic acest lucru.

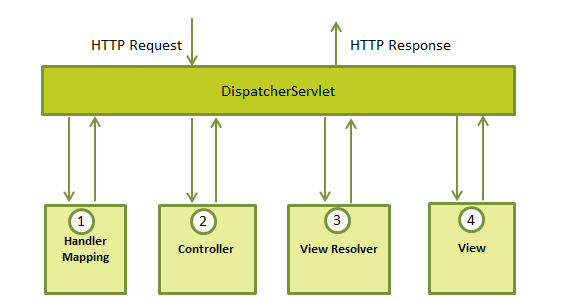


Figura 5. Fluxul unui request într-o aplicație MVC