Hîncu Alice-Ramona
Sisteme Distribuite în Internet
Grupa 244

Instalarea și configurarea serverelor de aplicații

Despre proiect

Proiectul propus este o aplicație web pentru gestionarea unui adăpost de animale, utilizând servlet-uri Java pentru diferite funcționalități, inclusiv vizualizarea animalelor și adăugarea sau adoptarea acestora. Arhitectura aplicației este structurată în jurul a trei servlet-uri principale și un listener pentru contextul aplicației.

S-au creat trei servlet-uri în cadrul aplicației:

- **AnimalManagementServlet** - gestionează adăugarea de noi animale și posibilitatea de a adopta un animal aleatoriu. Metode:
  - *doGet(HttpServletRequest, HttpServletResponse):* Afișează formularul pentru adăugarea unui animal și butonul de adoptare.
  - *doPost(HttpServletRequest, HttpServletResponse)*: Procesează datele de intrare și execută acțiuni în funcție de butonul apăsat (adăugare sau adoptare).
  - *validateInput(String, String, String)*: Validează intrările pentru nume, vârstă și rasă. Validările presupun: câmpul de nume, de rasă și de vârstă să nu fie goale, iar vârsta animalului să nu fie negativă și trebuie să fie de tip întreg
- **ShelterServlet** - afișează lista de animale din adăpost. Metode:
  - *doGet(HttpServletRequest, HttpServletResponse)*: Afișează animalele stocate în contextul aplicației.
- **HomePageServlet** - pagina principală care oferă linkuri către celelalte două servlet-uri. Metode:
  - *doGet(HttpServletRequest, HttpServletResponse)*: Afișează mesajul de bun venit și linkurile.

Listenerul utilizat în aplicația de gestionare a adăpostului de animale este un ServletContextListener, care este un tip special de listener în Java Servlet API. Acesta oferă un mecanism pentru a intercepta evenimentele de ciclu de viață ale întregii aplicații web, adică la

inițializarea și închiderea contextului aplicației. În cadrul acestui proiect, **AppContextListener** este utilizat pentru a inițializa datele aplicației la pornire. La pornirea aplicației, AppContextListener populează automat lista de animale cu cinci intrări inițiale. Aceste date devin imediat accesibile servlet-urilor AnimalManagementServlet și ShelterServlet pentru a fi afișate sau modificate. AppContextListener nu este destinat să intervină în procesarea cererilor HTTP individuale sau în gestionarea sesiunilor utilizatorilor. Rolul său este strict limitat la evenimentele de ciclu de viață ale aplicației în ansamblu.

Pentru animale, s-a creat clasa *Animal* care conține 3 atribute:

- *name* – numele animalului, de tip String
- *age* – vârsta animalului, de tip Integer
- *breed* – rasa animalului, de tip String

Request-uri HTTP

Request-uri HTTP relevante:

- **GET /home** - Afișează pagina principală a aplicației. Aceasta conține linkuri către celelalte servlet-uri.
- **GET /shelter** - Afișează lista de animale din adăpost. Accesează lista de animale din ServletContext și o prezintă utilizatorului.
- **GET /manage-animals** - Afișează formularul pentru adăugarea unui nou animal și butonul pentru adoptarea unui animal. Este interfața principală pentru gestionarea animalelor.
- **POST /manage-animals**
    - cu acțiunea "Add animal" și parametri: name, age, breed - Procesează adăugarea unui nou animal. Validează datele de intrare și, dacă sunt corecte, adaugă animalul în lista din ServletContext.
    - cu acțiunea "Adopt" - Procesează adoptarea unui animal aleatoriu. Alege un animal la întâmplare din lista din ServletContext și îl elimină, notificând utilizatorul despre adoptarea efectuată.

Exemple de utilizare

- Vizualizarea Adăpostului de Animale
    - Utilizatorul accesează HomePageServlet (e.g., /home).
    - Selectează linkul către ShelterServlet pentru a vedea animalele.
    - ShelterServlet afișează lista de animale.

      Shelter

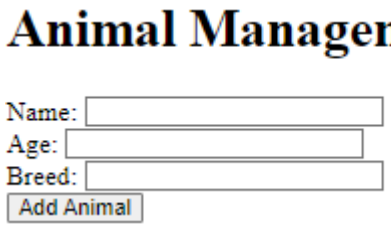
      Max - Labrador - 3 years old

      Charlie - Bulldog - 4 years old

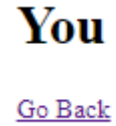
      Lucy - Poodle - 5 years old

      Daisy - Boxer - 1 years old
    -
- Adăugarea unui Animal Nou
    - Utilizatorul accesează HomePageServlet (e.g., /home).

- De pe pagina principală, utilizatorul accesează AnimalManagementServlet.
- Completează formularul pentru a adăuga un nou animal şi apasă "Add Animal".
- Dacă datele sunt valide, animalul este adăugat la listă şi utilizatorul este informat.

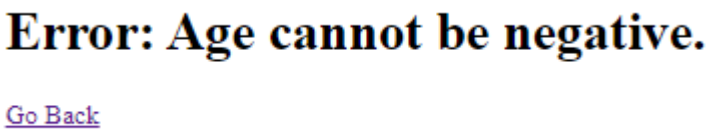


- Adoptarea unui Animal
  - În AnimalManagementServlet, utilizatorul apasă pe butonul "Adopt".
  - Un animal aleatoriu este eliminat din listă.
  - Utilizatorul este notificat cu privire la animalul adoptat prin afişarea numelui.



Contraexemplu de Utilizare:

- Introducerea unei vârste negative pentru un animal nou în AnimalManagementServlet va rezulta într-un mesaj de eroare şi refuzul adăugării animalului în listă.



Aplicaţia a fost dezvoltată pe sistemul de operare Windows, folosind JDK 11 şi Maven ca build tool. Proiectul este de tip JakartaEE 9.1 (Servlets 5), dar se foloseste si javax pentru tomcat-embedded.

Instalarea si configurarea tool-urilor necesare

Sistemul de operare folosit este Windows 10. Prin urmare am instalat si folosit:

- **Kitul de dezvoltare Java: JDK 11**
  - versiunea de Windows poate fi obținută de la adresa https://www.oracle.com/ro/java/technologies/javase/jdk11-archive-downloads.html
  - Arhiva am extras-o in locatia *C:\Java.* Am ales sa fac un folder special pentru java, deoarece am mai multe versiuni de java, toate pastrate acolo
  - In System Variables am adaugat 2 variabile:
    - *JAVA_HOME_11=C:\Java\ jdk-11.0.8.10-hotspot*
    - *JAVA_HOME=%JAVA_HOME_17%*

- M-am asigurat ca in PATH-ul din System Variables, am adaugat pe prima linie *%JAVA_HOME%\bin* . Fiind pe prima linie, are prioritate fata de alte java.exe-uri existente (cum ar fi cel din system32).

Urmează instalarea şi configurarea serverelor de aplicaţii:

- **Tomcat**
  - Poate fi obţinut de la adresa https://tomcat.apache.org/download-10.cgi , optiunea 64-bit Windows zip
  - Fişierul se dezarvihează şi se alege un loc pentru el care va fi folosit pentru System Variables
  - In System Variables am adaugat 2 variabile:
    - *CATALINA_HOME_10=C:\Tools\apache-tomcat-10.1.16*
    - *CATALINA _HOME=%CATALINA_HOME_10%*
- **Jetty**
  - Poate fi obţinut de la adresa https://repo1.maven.org/maven2/org/eclipse/jetty/jetty-home/11.0.12/jetty-home11.0.12.zip
  - Fişierul se dezarvihează şi se alege un loc pentru el care va fi folosit pentru System Variables
  - In System Variables am adaugat 2 variabile:
    - *JETTY_HOME_11=C:\Tools\jetty-home-11.0.12*
    - *JETTY_HOME=%JETTY_HOME_11%*
  - Se creează un director cu un numele *jetty-base,* într-o locaţie diferită de JETTY_HOME, şi după aceea se adaugă o variabilă de sistem:
    - JETTY_BASE= *C:\Tools\jetty-base*
  - În JETTY_BASE, trebuie activat modulul de deploy, care va face deploy automat la aplicaţii puse în interiorul directorului webapps. Se poate face asta rulând următoarea comandă:
    - java -jar %JETTY_HOME%/start.jar --add-module=deploy,http,jsp
- **Wildfly**
  - Poate fi obtinut de la adresa https://www.wildfly.org/downloads/ varianta 23.0.0 Preview EE 9 Distribution
  - In System Variables am adaugat 2 variabile:
    - *WILDFLY_HOME_23=C:\Tools\wildfly-23.0.2.Final*
    - *WILDFLY_HOME=%WILDFLY_HOME_23%*
- **GLASSFISH**
  - Poate fi obtinut de la adresa https://projects.eclipse.org/projects/ee4j.glassfish/downloads, Eclipse GlassFish 6.2.5, Full Profile
  - In System Variables am adaugat 2 variabile:
    - *GLASSFISH_HOME_6=C:\Tools\glassfish6*
    - *GLASSFISH_HOME=% GLASSFISH_HOME_6%*

Pentru IntelliJ, acesta a fost setupul:

- https://www.jetbrains.com/help/idea/creating-and-running-your-first-java-ee-application.html#new_project

Pentru toate cele 8 porniri, am făcut un script shell care să le pornească numit start-servlets

Deploy cu WAR

În primul rând, click dreapta pe fişierul pom şi apoi selectată opţiunea Maven > Reload Project. Obţinerea unei arhive WAR se poate realiza prin executarea comenzii mvn install. În urma execuţiei acestei comenzi se va obţine o arhivă WAR ce poate fi localizată în PROJECT_ROOT\target\servlets.war, unde PROJECT_ROOT este locaţia directorului proiectului.

TOMCAT:

- Se copiază arhiva WAR în locaţia %CATALINA_HOME%/webapps
- Se execută comanda %CATALINA_HOME%/bin/startup.bat
- În urma executării celor doi paşi menţionaţi anteriori, server-ul Tomcat va porni şi va servi servlet-urile din arhiva WAR la adresa http://localhost:8080/servlets/home
- Pentru oprire se execută comanda %CATALINA_HOME%/bin/shutdown.bat

JETTY

- Se copiază arhiva WAR în locaţia %JETTY_BASE%/webapps
- Se execută comanda java -jar %JETTY_HOME%/start.jar în locaţia %JETTY_BASE%
- În urma executării celor doi paşi menţionaţi anteriori, server-ul Jetty va porni şi va servi servlet-urile din arhiva WAR la adresa http://localhost:8080/servlets/home
- Ctrl+C pentru oprire, apoi Y pentru confirmare

WILDFLY

- Se copiază arhiva WAR în locaţia %WILDFLY_HOME%/standalone/deployments
  - Sau se poate intra pe http://localhost:9990/console/index.html si sa se dea upload la deploy (după ce s-a creat user-ul cu add-user.bat)
- Se execută comanda %WILDFLY_HOME%/bin/standalone.bat
- În urma executării celor doi paşi menţionaţi anteriori, server-ul WildFly va porni şi va servi servlet-urile din arhiva WAR la adresa http://localhost:8080/servlets/home
- Ctrl+C pentru oprire, apoi Y pentru confirmare

GLASSFISH

- Se execută comanda %GLASSFISH_HOME%/bin/asadmin start-domain pentru a porni server-ul GlassFish. (de preferat din target unde este war-ul pentru a fi mai uşoară următoarea comandă)
- Se execută comanda %GLASSFISH_HOME%/bin/asadmin deploy servlets-war-2-1.0-SNAPSHOT.war start-domain în locaţia unde se află arhiva WAR, sau se dă path-ul către WAR dacă executăm comanda din alt director.
  - cd C:\Users\alice.hincu\Desktop\stickFacultate\Master_SDI_An1 (1)\anu1\tpjad\servlets-war-2\target
- servlet-urile din arhiva WAR la adresa http://localhost:8080/servlets/home
- Pentru oprire: asadmin stop-domain domain1

Hîncu Alice-Ramona
Sisteme Distribuite în Internet
Grupa 244

Deploy cu context extern, fără arhive WAR

Deploy-ul cu context extern constă, spre deosebire de deploy-ul cu arhive WAR, în a preciza, într-un fişier de configurare oferit AS-ului, locaţia unde poate fi găsită aplicaţia despachetată

In Maven, pentru a crea o structură de directoare a aplicaţiei web nemîpachetate ("Exploding the WAR") se poate folosi pluginul maven-war-plugin prin configurarea unei faze care să extragă conţinutul WAR-ului după ce a fost construit.

```
<plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-war-plugin</artifactId>
        <version>3.3.1</version>
        <configuration>
                <!-- Specificaţi calea pentru exploded WAR -->

                <webappDirectory>${project.build.directory}/exploded</webappDirectory>
        </configuration>

</plugin>
```

După aceea se execută comanda mvn clean package, iar aplicaţia poate fi localizată în directorul PROJECT_ROOT\target\exploded, unde PROJECT_ROOT este calea absolută către directorul unde este desfăşurată aplicaţia.

În continuare vom detalia paşii necesari deploy-ului acestei aplicaţii pentru fiecare AS-urile Tomcat şi Jetty.

TOMCAT

- În directorul %CATALINA_HOME%/conf/Catalina/localhost se va crea un fişier cu numele servlets_extern_tomcat.xml cu următorul conţinut:

  ```
  <Context

  displayName="servlets_extern_tomcat"

  docBase= "PROJECT_ROOT/build/exploded"

  reloadable= "true"

  />
  ```

- Se execută comanda %CATALINA_HOME%/bin/startup.bat
- În urma executării celor doi paşi menţionaţi anteriori, server-ul Tomcat va porni şi va servi servlet-urile la adresa http://localhost:8080/servlets_extern/home

JETTY

- În directorul creat la punctul 1.2 din secţiunea despre deploy cu arhive WAR, jetty_base/webapps, se va crea un fişier cu un nume arbitrar, fie servlets_extern_jetty.xml cu următorul conţinut

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE Configure PUBLIC "-//Jetty//Configure//EN"
"https://www.eclipse.org/jetty/configure_10_0.dtd">

<Configure class="org.eclipse.jetty.webapp.WebAppContext">

<Set name="contextPath">/servlets_extern_jetty</Set>

<Set name="war">PROJECT_ROOT/target/exploded</Set>

</Configure>
```

- Se execută comanda java -jar %JETTY_HOME%/start.jar în locația jetty_base
- În urma executării celor doi pași menționați anteriori, server-ul Jetty va porni și va servi servlet-urile la adresa http://localhost:8080/servlets_extern/home.

Aplicatii embedded

Implementarea unor aplicații embedded constă în a folosi AS-uri embedded, cărora le sunt furnizate programatic un set de servlet-uri pe care să le servească. Spre deosebire de modalitățile anterioare de deploy, în acest caz aplicația va fi sub forma unui jar executabil, ce poate fi creat cu ajutorullui plugin-ului maven-assembly-plugin

În urma executării comenzii mvn clean compile assembly:single se va genera un jar ce conține aplicația embedded, care se află în locația PROJECT_ROOT/target/servlets-tomcat-embedded-2-jar-with-dependencies.jar, care poate fi executat folosind comanda java -jar servlets-tomcat-embedded-2-jar-with-dependencies.jar din locația unde se află jar-ul, sau se dă path-ul către jar dacă se execută comanda din alt director. În continuare vom prezenta succint modificările necesare pentru implementarea aplicațiilor embedded folosind AS-urile Tomcat și Jetty.

Embedded Tomcat

- A fost necesară specificarea dependinței org.apache.tomcat.embed în fișierul pom.xml
- Schimbările de cod au presupus instanțierea unui obiect de tip Tomcat, căruia i s-au furnizat servlet-urile și path-urile aferent fiecăruia, și pornirea server-ului
- După rularea jar-ului executabil, aplicația poate fi accesată la adresa http://localhost:8080/home

Embedded Jetty

- A fost necesară specificarea dependinței org.eclipse.jetty în fișierul pom.xml.
- Schimbările de cod au presupus instanțierea unui obiect de tip Server, căruia i s-au furnizat servlet-urile și path-urile aferent fiecăruia, și pornirea server-ului.
- După rularea jar-ului executabil, aplicația poate fi accesată la adresa http://localhost:8080/home