MANAGEMENTUL ACTIVITATILOR SPORTIVE DIN CADRUL UPT

Candidat: Antonyo Topolniceanu

Coordonator științific: Conf.dr.ing. Ciprian-Bogdan Chirilă

Timișoara

Septembrie 2023

Cuprins

ABSTRACT

1 INTRODUCERE

2 TEHNOLOGII WEB

2.1 Java

Java a fost creat de către o echipă de cercetători condusă de către James Gosling, la Sun Microsystems, pentru a facilita comunicarea dintre dispozitivele electronice de consum. Edificarea limbajului de programare Java a început în anul 1991 și, în scurt timp, concentrarea echipei s-a schimbat pe o nouă nișă, World Wide Web. Java a fost lansat pentru prima dată în 1995, iar capacitățile acestuia de a oferi interactivitate și control multimedia a arătat că este deosebit de potrivit pentru Web.

Până la sfârșitul anilor 1990, Java a adus multimedia pe internet și a început să se extindă dincolo de Web, alimentând dispozitive de consum, computere, chiar și computerul de bord al roverelor de explorare pe Marte ale NASA. Datorită popularității, Sun Microsystems a creat diferite varietăți de Java pentru diferite scopuri, inclusiv Java SE pentru computerele de consum, Java ME pentru dispozitivele incorporate și Java EE pentru servere de internet și supercomputere.

Diferența dintre modul de funcționare Java și alte limbaje de programare a fost revoluționară. Codul în alte limbi este mai întâi tradus de un compilator în instrucțiuni pentru un anumit tip de computer. În schimb, compilatorul Java transformă codul în Bytecode, care este apoi interpretat de softwareul numit Java Runtime Environment (JRE) sau mașină virtuală Java. JRE acționează ca un computer virtual care interpretează Bytecode și îl traduce pentru computerul gazdă. Din acest motiv, codul Java poate fi scris în același mod pentru mai multe platforme ("Write Once, Run Everywhere"), ceea ce a contribuit la popularitatea sa pentru utilizarea pe Internet, unde multe tipuri diferite de computere pot prelua aceeași pagină Web.

În ciuda asemănării numelor, limbajul JavaScript care a fost conceput pentru a rula în browserele web nu face parte din Java. Inițial se numea Mocha și apoi LiveScript înainte ca Netscape să primească o licență de marketing de la Sun Microsystems.

2.2 Javascript

Dezvoltat de Netscape în colaborare cu Sun Microsystems, JavaScript este una dintre cele mai populare tehnologii de bază ale web. De la începuturile sale, a fost o parte integrantă a aplicațiilor web, realizând interactivitatea și dinamica acestora.

Începând cu 2023, 98.7% dintre site-uri folosesc Javascript în partea clientului pentru prelucrarea comportamentului paginilor web, deseori încorporând librării third-party. Toate browserele web majore au un interpretor JavaScript dedicat pentru a executa codul pe dispozitivele utilizatorilor.

Este un limbaj de nivel înalt, adesea compilat just-in-time, la run-time, care se conformează standardului ECMAScript. Este un limbaj de programare orientat pe obiecte bazate pe prototip și funcții de primă clasă. Include tipuri dinamice. Este multi-paradigmă și acceptă stiluri de programare bazate pe evenimente, funcționale și imperative. Are interfețe de programare a aplicațiilor pentru lucrul cu text, date, expresii regulate, structuri de date și manipulare de Document Object Model.

ECMAScript este un standard pentru limbajele de scriptare, incluzând JavaScript JScript și ActionScript. De asemenea este cunoscut ca un standard care să asigure interoperabilitatea paginilor web în diferite browsere.

Interpretoarele JavaScript au fost utilizate inițial doar în browserele web, dar acum sunt componente de bază ale unor servere și ale unor varietăți de aplicații, cel mai popular sistem de rulare folosit în aceste scopuri fiind Node.js.

Weak typing-ul limbajului de programare JavaScript poate creea foarte ușor erori. În alte limbaje, cum ar fi Java și C++, trebuie specificat în mod explicit tipul unui variabile. Cu toate acestea, în JavaScript, interpretorul deduce automat tipul de date al unei variabile pe baza valorii care îi este atribuită.

Constrângerea agresivă de tip este o caracteristică a JavaScript care obligă diferitele tipuri de date să fie compatibile între ele. De exemplu dacă încercați să atribuiți un număr unei variabile de tip string, JavaScript va converti automatul numărul într-un string, indiferent de valoarea inițială. Prin urmare, acest lucru poate duce la rezultate neașteptate și poate fi dificil de depanat.

Din cauza acestor deficiențe, pentru dezvoltarea de aplicații pe scară largă, Microsoft a creat TypeScript.

2.3 Typescript

Prin definiție, „TypeScript este JavaScript pentru dezvoltarea de aplicații scalabile".

TypeScript este un limbaj puternic tipizat, orientat pe obiecte, compilat. A fost proiectat de Anders Hejlsberg la Microsoft. TypeScript este atât un limbaj, cât și un set de instrumente. TypeScript este un superset tip de JavaScript compilat în JavaScript. Cu alte cuvinte, TypeScript este JavaScript plus câteva caracteristici suplimentare.

A fost lansat publicului în octombrie 2012, cu versiunea 0.8, după doi ani de dezvoltare internă la Microsoft. La scurt timp după lansarea publică inițială, limbajul în sine a fost lăudat, dar s-a criticat lipsa suportului matur al multor IDE, în afară de Microsoft Visual Studio, care nu era disponibil pe Linux și OS X la acel moment. Din aprilie 2021, există suport în alte IDE-uri și editori de text, inclusiv Emacs, Vim, WebStorm, Atom și Visual Studio Code al Microsoft.

TypeScript este portabil peste browsere, dispozitive și sisteme de operare. Poate rula în orice mediu în care rulează JavaScript. Spre deosebire de omologii săi, TypeScript nu are nevoie de un VM dedicat sau de un mediu de rulare specific pentru a fi executat.

Este superior celorlalți omologi ai săi, cum ar fi limbajele de programare CoffeeScript și Dart, într-un mod în care TypeScript este JavaScript extins. În schimb, limbaje precum Dart, CoffeeScript sunt limbaje noi în sine și necesită un mediu de execuție specific limbii.

JavaScript este un limbaj interpretat. Prin urmare, trebuie rulat pentru a testa dacă este valid, ceea ce poate duce la ore întregi de debugging. Transpilerul TypeScript oferă caracteristica de verificare a erorilor. TypeScript va compila codul și va genera erori de compilare, dacă găsește erori de sintaxă. Acest lucru ajută la evidențierea erorilor înainte de rularea codului.

JavaScript nu este puternic tipizat. TypeScript vine cu un sistem opțional de tipizare statică și inferență de tip prin Serviciul de limbaj TypeScript, denumit TLS. Tipul unei variabile, declarată fără tip, poate fi dedus de TLS pe baza valorii sale.

2.4 Angular

Angular este o platformă de dezvoltare software, construită pe TypeScript. Că platformă, Angular are un cadru bazat pe component pentru construirea de aplicații web scalabile, o colecție de biblioteci bine integrate care acoperă o mare varietate de funcții, inclusiv rutare, gestionare a formularelor, comunicare client-server și multe altele. De asemenea, are o suită de instrumente pentru developeri, care facilitează dezvoltarea, construirea, testarea și actualizarea codului.

Ecosistemul Angular este format dintr-un grup divers de peste 1,7 milioane de developeri, autori de biblioteci și creatori de conținut.

2.4.1 Componente

Componentele sunt blocurile principale care compun o aplicație. O componentă include o clasă TypeScript cu un decorator @Component(), un șablon HTML și stiluri CSS. Decoratorul @Component() specifică următoarele informații:

- Un selector CSS care definește modul în care componenta este utilizată într-un șablon de HTML. Elementele HTML din șablon care se potrivesc cu acest selector devin instanțe ale componentei.
- Un șablon HTML care dictează la Angular cum să randeze componenta.
- Un set opțional de stiluri CSS care definesc aspectul elementelor HTML ale șablonului.

Următoarea este o componentă Angular minimală:

```
//hello-world.component.ts
import { Component } from '@angular/core';

@Component({
    selector: 'hello-world',
    template: `
        <h2>Hello World</h2>
        <p>This is my first component!</p>
    `
})
export class HelloWorldComponent {
    // Codul ce descrie comportamentul componentei.
}
```

Pentru a utiliza această componentă, trebuie adăugat selectorul componentei într-un șablon:

```
<hello-world></hello-world>
```

Când Angular randează această componentă, DOM-ul rezultat arată astfel:

```
<hello-world>
    <h2>Hello World</h2>
    <p>This is my first component!</p>
</hello-world>
```

2.4.2 Șabloane

Fiecare componentă are un șablon HTML care declară cum se randează acea componentă. Definirea acestui șablon se poate face fie în linie, fie într-un fiser separat.

Angular adaugă elemente de sintaxă care extind HTML, pentru a facilita inserarea de valori dinamice din componenta Angular. Actualizează automat DOM-ul atunci când starea componentei se schimbă. O aplicație a acestei caracteristici este inserarea de text dinamic, așa cum se arată în exemplul următor.

```
//hello-world-interpolation.component.html
<p>{{ message }}</p>
```

Valoarea mesajului provine din clasa de componente:

```
//hello-world-interpolation.component.ts
import { Component } from '@angular/core';

@Component ({
selector: 'hello-world-interpolation',
templateUrl: './hello-world-interpolation.component.html'
})
export class HelloWorldInterpolationComponent {
    message = 'Hello, World!';
}
```

Când aplicația încarcă componenta și șablonul acesteia, utilizatorul vede următoarele:

```
//hello-world-interpolation.component.html
<p>Hello, World!</p>
```

Acoladele duble îi indică lui Angular să interpoleze conținutul din ele.

Angular acceptă, de asemenea, legături de proprietăți, pentru a facilita transmiterea valorilor dintre componente și pentru a seta diferite proprietăți.

```
<p
    [id]="sayHelloId"
    [style.color]="fontColor">
    Se seteaza fontColor in clasa typescript al componentei
</p>
```

Utilizarea parantezelor drepte indică legătura dintre proprietatea sau atributul la o valoare din clasa componentelor.

Pentru a asculta acțiunile utilizatorului, cum ar fi apăsările de taste, mișcările mouseului, clickurile, se definesc ascultători de evenimente, specificând numele evenimentului între paranteze rotunde:

```
<button
    (click)="sayMessage()">
    functia sayMessage este definita in clasa typescript al componentei
</button>
```

2.4.3 Dependency Injection

Dependency Injection este partea din cadrul Angular care oferă componentelor acces la servicii și alte resurse. Angular oferă posibilitatea de a injecta un serviciu într-o componentă pentru a oferi acelei componente acces la serviciu.

Decoratorul @Injectable() definește o clasă ca serviciu în Angular și îi permite lui Angular să o injecteze într-o componentă ca dependență. De asemenea, decoratorul @Injectable() indică faptul că o componentă, clasă sau NgModule are o dependență de un serviciu.

Injectorul este mecanismul principal. Angular creează un injector la nivel de aplicație în timpul procesului de bootstrap și injectoare suplimentare după cum este necesar.

Un injector creează dependențe și menține un container de instanțe de dependență pe care le reutilizează, dacă este posibil. Un furnizor este un obiect care îi spune unui injector cum să obțină sau să creeze o dependență.

2.4.4 Servicii

Serviciul este o categorie largă care cuprinde orice valoare, funcție sau caracteristică de care are nevoie o aplicație. Un serviciu este de obicei o clasă cu un scop restrâns si bine definit. Angular stie sa distinga componentele de servicii pentru a creste modularitatea si reutilizarea.

Im mod ideal, sarcina unei componente este sa se ocupe doar de experienta utilizatorului. O componenta ar trebui sa prezinte proprietati si metode pentru legarea datelor pentru a media intre vizualizare si logica aplicatiei.

Serviciile sunt bune pentru sarcini precum preluarea datelor de pe server, validarea intrărilor utilizatorului sau conectarea direct la consolă. Prin definirea unor astfel de sarcini de procesare într-o clasă de servicii injectabilă, acele sarcini devin disponibile oricărei componente. De asemenea, pentru a face aplicația mai adaptabilă, se pot injecta furnizori diferiți ai aceluiași tip de serviciu, după caz, în circumstanțe diferite.

În Angular, Dependency Injection face aceste servicii disponibile pentru componente.

2.4.5 Module

Aplicațiile Angular sunt modulare și Angular are propriul său sistem de modularitate numit NgModules. NgModules sunt containere pentru un bloc coeziv de cod dedicat unui domeniu de aplicație, unui flux de lucru sau unui set de capabilități strâns legate. Ele pot conține componente, furnizori de servicii și alte fișiere de cod al căror domeniu este definit de NgModule care le conține. Ei pot importa funcționalități care sunt exportate din alte NgModules și pot exporta funcționalitatea selectată pentru a fi utilizate de alte NgModules.

Fiecare aplicație Angular are cel puțin o clasă NgModule, modulul rădăcină, care este denumit în mod convențional AppModule și rezidă într-un fișier numit app.module.ts.

2.5 Spring Boot

2.6 Spring Data JPA

2.7 PostgreSQL

3 PROIECTAREA

4 IMPLEMENTARE

5 TESTARE SI INSTALARE

6 CONCLUZII

BIBLIOGRAFIE