

APLICATII WEB CU SUPORT JAVA

Prof.Dr.Ing. Liliana Dobrică

*Universitatea POLITEHNICA Bucuresti
Facultatea Automatica si Calculatoare*

Aplicatii Web cu Suport Java, sem.I, 2024-2025

1

1

Agenda cursului

• Java

- equals()
- hashCode()
- Java Generics
- Aplicatii Spring Boot

Aplicatii Web cu Suport Java, sem.I, 2024-2025

2

2

Metoda equals() din Object

Metoda `equals()` din `Object` va compara adresele de memorie

```
public boolean equals(Object obj) {  
    return (this == obj);  
}
```

Implementarea unei clase fara metoda `equals()`. Java se va baza pe implementarea superclasei, Se poate ajunge la metoda `equals()` din `Object`

Pentru relatia de mostenire

```
class Persoana {  
    private String nume;  
    public boolean equals(Object o) {  
        if(o == this)  
            return true;  
        if(o == null || this.getClass() != o.getClass())  
            return false;  
        Persoana p = (Persoana) o;  
        return Objects.equals(this.nume, p.nume);  
    }  
    ...  
}
```

Pentru relatia de mostenire

```
class Student extends Persoana {  
    private int nrMatricol;  
    public boolean equals(Object o) {  
        if(o == this)  
            return true;  
        if(o == null || this.getClass() != o.getClass())  
            return false;  
        Student s = (Student) o;  
        return this.nrMatricol == s.nrMatricol  
            && super.equals(s);  
    }  
    ...  
}
```

Aplicatii Web cu Suport Java, sem.I, 2024-2025

5

5

“Metoda equals() implementeaza o relatia de echivalenta intre obiecte non-null.”

- Reflexiva
 - `x.equals(x) ==true`
- Simetrica
 - `x.equals(y)==y.equals(x)`
- Tranzitiva
 - `if ((x.equals(y)&& x.equals(z)) atunci y.equals(z) trebuie sa fie true`
- Consistenta

Aplicatii Web cu Suport Java, sem.I, 2024-2025

6

6

Relatia de mostenire (1)

exemplu utilizarea operatorului instanceof nu este o solutie buna!

```
class Vehicol {  
    private String numeModel;  
    public boolean equals(Object o) {  
        if(o == this)  
            return true;  
        if(! (o instanceof Vehicol))  
            return false;  
        Vehicol v = (Vehicol) o;  
        return Objects.equals(  
            this.numeModel, v.numeModel);  
    }  
    ...  
}
```

Aplicatii Web cu Suport Java, sem.I, 2024-2025

7

Relatia de mostenire (2)

exemplu instanceof nu este o solutie buna!

```
class Tren extends Vehicol {  
    public boolean equals(Object o) {  
        if(o == this)  
            return true;  
        if(! (o instanceof Tren))  
            return false;  
        return super.equals(o);  
    }  
    ...  
}
```

Aplicatii Web cu Suport Java, sem.I, 2024-2025

8

8

Relatia de mostenire (3) exemplu instanceof nu este o solutie buna!

```
Vehicol vehicol1 = new Vehicol("ABC");
Tren vehicol2 = new Tren("ABC");
vehicol1.equals(vehicol2); // true
vehicol2.equals(vehicol1); // false
```

Tren este instanceof Vehicol
Dar Vehicol nu este instanceof Tren

**Hashmap
HashCode
HashSet
Hashtable**

The screenshot shows the JavaDoc for the `hashCode()` method. It includes the method signature, a brief description, the general contract, implementation requirements, and related methods.

hashCode

```
public int hashCode()
```

Returns a hash code value for the object. This method is supported for the benefit of hash tables such as those provided by `HashMap`.

The general contract of `hashCode` is:

- Whenever it is invoked on the same object more than once during an execution of a Java application, the `hashCode` method must consistently return the same integer, provided no information used in `equals`'s comparisons on the object is modified. This integer need not remain consistent from one execution of an application to another execution of the same application.
- If two objects are equal according to the `equals` method, then calling the `hashCode` method on each of the two objects must produce the same integer result.
- It is *not* required that if two objects are unequal according to the `equals` method, then calling the `hashCode` method on each of the two objects must produce distinct integer results. However, the programmer should be aware that producing distinct integer results for unequal objects may improve the performance of hash tables.

Implementation Requirements:

As far as is reasonably practical, the `hashCode` method defined by class `Object` returns distinct integers for distinct objects.

Returns:
a hash code value for this object.

See Also:
`equals(java.lang.Object)`,
`System.identityHashCode(java.lang.Object)`

Aplicatii Web cu Suport Java, sem.I, 2024-2025

11

11

The slide contains three bolded section titles: **Hashmap**, **HashSet**, and **Hashtable**. Below them is a text block and a statement about the `hashCode` method.

Hashmap
HashSet
Hashtable

Reprezinta structuri care folosesc `hashCode` – uri pentru memorarea si obtinerea rapida a obiectelor Objects

In Java metoda `hashCode ()` returneaza un `int`

Aplicatii Web cu Suport Java, sem.I, 2024-2025

12

12

Detaliere

Pp ca avem un array foarte mare de elemente de tip Persoana

Caut dupa nume o persoana

In mod normal se face loop in array si folosesc metoda equals()

Este bine, dar dureaza mai mult

| Index | Continut |
|-------|----------------------------------|
| 0 | Persoana("Liliana", Adresa(...)) |
| 1 | Persoana("Iulia", Adresa(...)) |
| ... | ... |
| 34 | Persoana("Luca", Adresa(...)) |
| ... | ... |
| 998 | Persoana("Radu", Adresa(...)) |

Detaliere

Solutie – se utilizeaza hashCode()

```
@Override  
public int hashCode() {  
    int result = nume.hashCode();  
    result = 31 * result + adresa.hashCode();  
    return result;  
}
```

603287366

603287366 este mare in
comparatie cu dimensiunea array-
ului ex. 999

| Index | Continut |
|-------|----------------------------------|
| 0 | Persoana("Liliana", Adresa(...)) |
| 1 | Persoana("Iulia", Adresa(...)) |
| ... | ... |
| 34 | Persoana("Luca", Adresa(...)) |
| ... | ... |
| 998 | Persoana("Radu", Adresa(...)) |

Detaliere

Solutie:

- Există o **functie de compresie** pentru array-uri care poate să facă o conversie de hash-code într-un anumit index
- Combinarea dintre **hash code** și **functia de compresie** se numește **functie hash**.
- Calcularea unui hash code și aplicarea funcției de compresie se realizează într-o durată constantă de timp.
- Astfel prin utilizarea de hash se îmbunătățește performanța pentru anumite operații.
- Un hash code bun este acela care poate să fie cât mai unic posibil!
 - Altfel incetineste executia!
- Trebuie să aibă aceeași valoare pentru două obiecte egale între ele!

Detaliere

Solutie:

Hash code unic (aproximativ)?

- Să includă toate atributele clasei
 - Toate atributele luate în considerare de metoda equals()
- Înmulțirile cu numere prime sunt de ajutor pentru a evita conflictele - să indice același index

Java Generics

Aplicatii Web cu Suport Java, sem.I, 2024-2025

17

17

Java Generics

Pentru Java inainte de Java 1.5, exemplul dat

Din compatibilitate este acceptat de compilator, cu warning

ArrayList fara a avea tipul specificat, memoreaza obiectele ca Object

```
ArrayList stringlist2 = new ArrayList();  
  
stringlist2.add(new String("Student 1"));  
stringlist2.add(new String("Student 2"));  
String element = (String) stringlist2.get(0);
```

Cast la String fara cand se obtine elementul

Aplicatii Web cu Suport Java, sem.I, 2024-2025

18

18

Java Generics

Este permis deoarece Student este subclasa Object

```
ArrayList stringlist2 = new ArrayList();
stringlist2.add(new String("Student 1", 1234));
stringlist2.add(new String("Student 2"));
String element = (String) stringlist2.get(0);
```

Va genera Exceptie la executie!

Java Generics

Solutie

- Utilizare generics
- Se va specifica pentru lista ca poate contine doar elemente de tip String

Doar elemente de tip String in array

```
ArrayList<String> stringlist2 = new ArrayList<String>();
stringlist2.add(new String("Student 1"));
stringlist2.add(new String("Student 2"));
String element = stringlist2.get(0);
```

Nu mai este necesar cast!

Java Generics

Clasa parametrizata

```
public class Cutie<T> {  
    private T t;  
  
    public void add(T t) {  
        this.t = t;  
    }  
  
    public T get() {  
        return t;  
    }  
}
```

T – de la Template

Orice tip de Object se
poate pune intr-o "Cutie"

Java Generics

Clasa parametrizata

```
public class Cutie<T> {  
    private T t;  
  
    public void add(T t) {  
        this.t = t;  
    }  
  
    public T get() {  
        return t;  
    }  
}
```

T – de la Template

Orice tip de Object se
poate pune intr-o "Cutie"

Dar tipuri primitive?

```
private A a;  
private A b;
```

Auto boxing

Toate tipurile primitive au clase wrapper

- Integer pentru int
- Character pentru char
- etc

Se poate crea ArrayList<Integer>

Prin autoboxing / -unboxing Java se ocupa automat de tipurile primitive

Aplicatii Web cu Suport Java, sem.I, 2024-2025

23

23

Pereche de parametrii

```
public class  
Pereche<A, B> {  
    private A a;  
    private B b;  
}
```

Utilizare

```
Pereche<String, Integer> p1 = new Pereche<>("abc", 1);
```

Sintaxa cu restrictie

```
public class Pereche<A, B extends Numar> {  
    private A a;  
    private B b;  
}
```

Aplicatii Web cu Suport Java, sem.I, 2024-2025

24

24

Aplicatii in Spring

- Introducere
- Modelul componentelor Spring
- Componerea sistemelor complexe
- Testarea aplicatiilor Spring cu Postman

Aplicatii Web cu Suport Java, sem.I, 2024-2025

25

25

Spring Initializer

- Introducere in Spring Boot

(1) Se acceseaza link-ul
start.spring.io

Gradle, Java,
Dependencies,
etc.

Maven, Java,
Dependencies,
etc.

(2) Dupa configurarea proiectului se
apasa butonul GENERATE

Aplicatii Web cu Suport Java, sem.I, 2024-2025

26

26

Spring Initializer

- Introducere in Spring Boot
- Exemplu dependente de luat in considerare pentru proiectul Java :

- **WEB: Spring Web** – pentru aplicatie SpringMVC. Foloseste si include ApacheTomcat
- **WEB: Jersey** – ofera suport JAX-RS pentru dezvoltarea serviciilor web REST
- **Template Engines: Tymeleaf** – ofera afisarea corecta in browser a informatiilor din HTML
- **SQL: Spring Data JPA** – ofera Java Persistence API pentru datele din Spring si Hibernate
- **SQL: H2 DataBase** – baza de date in-memorie cu suport de acces JDBC API . Ofere accesul din browser la consola
- **I/O: Validation** – validarea intrarilor si a iesirilor cf. JSR-303

Aplicatii Web cu Suport Java, sem.I, 2024-2025

27

27

(3) Se deschide proiectul generat in IntelliJ

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure for "demospringM2". It includes a "src" folder containing "main" and "test" subfolders, and a "resources" folder.
- Code Editor:** Displays the main file "DemospringM2Application.java" with the following code:

```
package upb.awy.demospringM2;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class DemoSpringM2Application {
    public static void main(String[] args) {
        SpringApplication.run(DemoSpringM2Application.class, args);
    }
}
```
- Run Tab:** Shows the command "Sync" was run at 10:53 AM on 9/18/2024.
- Problems Tab:** Shows "No problems in DemoSpringM2Application.java".

Aplicatii Web cu Suport Java, sem.I, 2024-2025

28

28

• Introducere in Spring Boot

(4) Se realizeaza Build

```

package upb.awj.demoSpringM2;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoSpringM2Application {

    public static void main(String[] args) {
        SpringApplication.run(DemoSpringM2Application.class, args);
    }
}

```

Aplicatii Web cu Suport Java, sem.I, 2024-2025

29

29

Controller Spring

(5) Se adauga proiectului clasa Controller

- Clasa este responsabila cu preluarea si tratarea cererilor HTTP

Clasa responsabila de
cereri HTTP

```

package upb.awj.demoSpringM2;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class UnController {

    @GetMapping("/")
    @ResponseBody
    public String index(){
        return "Hello world!";
    }
}

```

Care cerere? Cum raspunde?

Clasa responsabila de
cereri HTTP

@Controller face posibila inregistarea acestei clase in contextul aplicatiei
Spring

Aplicatii Web cu Suport Java, sem.I, 2024-2025

30

30

•15

Despre HTTP

Structura unei URL

http://acs.student.website.com://1234/alta/cale?param=valoare&p2=v2

Protocol Hostname Port Cale Parametrii

- Metode HTTP: GET, POST, PUT, DELETE,etc
- Mai multe detalii la
 - <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>
- Request / Response

Header Body

Aplicatii Web cu Suport Java, sem.I, 2024-2025

31

31

Diferite Căi și Variabile de Cale

@RequestMapping - Se înregistrează calea URL specifică a controllerului "/numit"

Aplicatia web trebuie să se execute oriunde "everywhere"

Alte adnotari care pot fi utilizate pentru a accesa detalii din request:
@RequestParam, @RequestBody sau @RequestHeader
 Recomandare de studiu: Documentatie Spring!

... si endpoint inclusiv metoda HTTP GET

Injectia cu dependenta (Spring dependency injection)

Aplicatii Web cu Suport Java, sem.I, 2024-2025

32

32

Componarea sistemelor complexe prin servicii

Spring se va ocupa de instantiere

```

package upb.awj.demoSpringM2;
import org.springframework.stereotype.Service;
@Service
public class ExtindereNameService {
    public String extindere(String in){
        if ("Lili".equals(in)){
            return "Liliana";
        }
        return in;
    }
}

```

Aplicatii Web cu Suport Java, sem.I, 2024-2025

33

33

Componarea sistemelor complexe prin servicii

Injectie dependenta

Injectie dependenta

Aplicatii Web cu Suport Java, sem.I, 2024-2025

34

34

Modelul de Componente Spring (=Injectie Dependentă)

- @Component
 - @Controller/ @RestController
 - @Repository
 - @Service
 - Spring va crea un obiect bean al clasei cu acelasi nume (numele obiectului va incepe cu litera mica!)
 - @Autowired //ApplicationContext.getBean(Class<?>)
-
- @Configuration
 - @Bean
 - @Scope(value=ConfigurableBeanFactory.X)

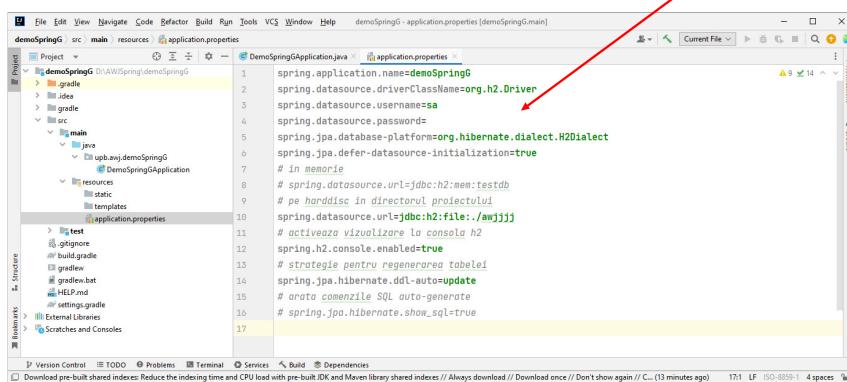
Aplicatii Web cu Suport Java, sem.I, 2024-2025

35

35

Accesul la baza de date: Configurare Spring

Se actualizeaza fisierul de proprietati



```
spring.application.name=demoSpringG
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.defer-datasource-initialization=true
# in memorie
# spring.datasource.url=jdbc:h2:mem:testdb
# pe harddisc in directorul proiectului
spring.datasource.url=jdbc:h2:file:/tmp/jjjjjj
# activeaza vizualizarea la consola H2
spring.h2.console.enabled=true
# strategie pentru regenerarea tabelor
spring.jpa.hibernate.ddl-auto=update
# creaza comenzile SQL auto-generate
# spring.jpa.hibernate.show_sql=true
```

Aplicatii Web cu Suport Java, sem.I, 2024-2025

36

36

Accesul la baza de date: Creare Entity si Repository

-

Adnotari JPA

Componenta Spring

```
@Entity  
public class Utilizator {  
    @Id  
    String nume;  
}
```

Tipul de entitate stocata

```
@Repository  
public interface UserRepository extends JpaRepository<Utilizator, String> {  
}
```

Tipul cheii primare

Aplicatii Web cu Suport Java, sem.I, 2024-2025

37

37

Accesul la baza de date: Creare Entity si Repository

```
@Repository  
public interface UserRepository extends JpaRepository<Utilizator, String> {  
}
```

- Prin extinderea `JpaRepository`, interfata `UserRepository` va mosteni 18 metode pentru a efectua operatiile obisnuite de persistenta
- Interfata `JpaRepository` este parametrizata cu 2 parametrii, tipul clasei cu care va lucra si tipul atributului ID
- Interfata este automat implementata la executia aplicatiei prin Spring Data

Aplicatii Web cu Suport Java, sem.I, 2024-2025

38

38

Accesul la baza de date: exemplu utilizare Repository

```

•
    • @Controller
      @RequestMapping("/numitEx")
      public class ControllerNumitExtins {
        private ExtindereNumeService ens;
        private RepositoryU utilizatori;

        public ControllerNumitExtins(ExtindereNumeService ens, RepositoryU ur){
          this.ens=ens;
          this.utilizatori=ur;
        }

        @GetMapping("/nume/{nume}")
        @ResponseBody
        public String numitEx(@PathVariable("nume") String nume){
          if(utilizatori.existsById(nume)){
            Utilizator u=new Utilizator();
            u.nume=nume;
            utilizatori.save(u);
          }
          return "Hello " + ens.extindere(nume)+ "!";
        }
      }
    
```

Injectie
Dependenta

Auto-generate

Aplicatii Web cu Suport Java, sem.I, 2024-2025

39

39

Accesul la baza de date: Utilizarea consolei pentru explorare

Se actualizeaza fisierul de proprietati

```

File Edit View Navigate Code refactor Build Run Tools VCS Window Help demoSpringG - application.properties [demoSpringG:main]
demoSpringG src main resources application.properties
Project demoSpringG D:\JAVA\Spring\demoSpringG
  > .gradle
  > .idea
  > .gradle
  > .git
  > main
    > java
      > upb.avj.demoSpringG
        > DemoSpringApplication
    > resources
      > static
      > templates
    > test
      > build.gradle
      > gradlew
      > gradlew.bat
      > HELP.md
      > settings.gradle
  > External Libraries
  > Scratches and Consoles
  > Structure
  > Version Control
  > TODO
  > Problems
  > Terminal
  > Services
  > Build
  > Dependencies
  > Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built JDK and Maven library shared indexes // Always download // Download once // Don't show again // C... (13 minutes ago)
  > 17:11 LF ISO-8859-1 4 spaces /m

1 spring.application.name=demoSpringG
2 spring.datasource.driverClassName=org.h2.Driver
3 spring.datasource.username=sa
4 spring.datasource.password=
5 spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
6 spring.jpa.deferDataSourceInitialization=true
7 # in memorie
8 # spring.datasource.url=jdbc:h2:mem:testdb
9 # pe harddisc in directorul proiectului
10 spring.datasource.url=jdbc:h2:file:/tmp/jjjjjj
11 # activeaza vizualizarea la consola H2
12 spring.h2.console.enabled=true
13 # strategie pentru regenerarea tabelor
14 spring.jpa.hibernate.ddl-auto=update
15 # creaza comenzi SQL auto-generate
16 # spring.jpa.hibernate.show_sql=true
17

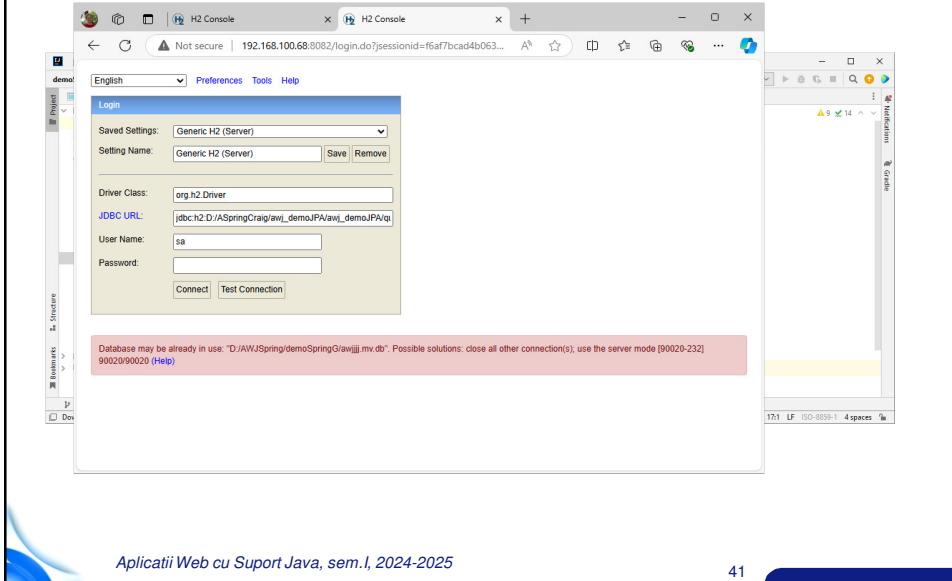
```

Aplicatii Web cu Suport Java, sem.I, 2024-2025

40

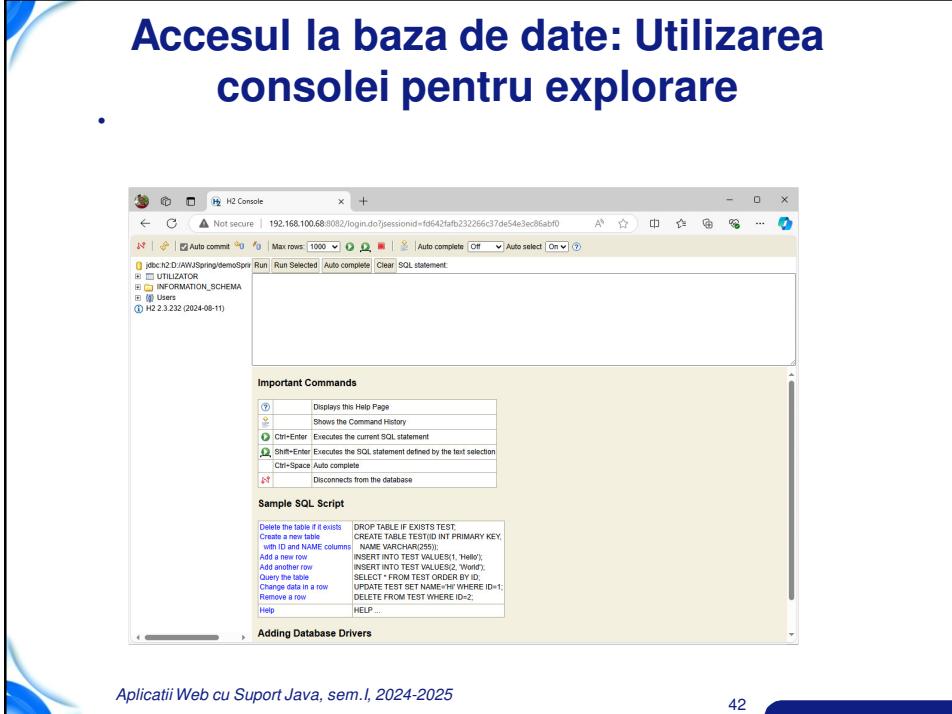
40

Accesul la baza de date: Utilizarea consolei pentru explorare



41

Accesul la baza de date: Utilizarea consolei pentru explorare



42

Accesul la baza de date: Utilizarea consolei pentru explorare

The screenshot shows the H2 Console interface. In the top right corner, there is a red box highlighting the word "iectati". The main area displays a SQL query: "SELECT * FROM UTILIZATOR;". Below the query, the results are shown in a table:

| NUME |
|--------|
| Anca |
| Daniel |
| Lili |
| Maria |
| Radu |

Below the table, it says "(5 rows, 3 ms)".

Aplicatii Web cu Suport Java, sem.I, 2024-2025

43

43

POSTMAN

Se foloseste pentru un GET

The screenshot shows the Postman interface. A red box highlights the text "Se foloseste pentru un GET". The main area shows a POST request to "http://localhost:8080/api/personal/". The "Body" tab is selected, showing the following JSON payload:

```
POST http://localhost:8080/api/personal/
{
  "name": "xxxx",
  "prenume": "yyyy"
}
```

The response pane shows a 200 OK status with the following JSON data:

```
{
  "name": "a",
  "prenume": "b",
  "id": 1,
  "name": "c",
  "prenume": "d",
  "id": 2
}
```

Aplicatii Web cu Suport Java, sem.I, 2024-2025

44

44

POSTMAN: depanare REST API

Se foloseste pentru un POST

The screenshot shows the Postman interface with a red box highlighting the 'Body' tab where JSON data is being sent. The JSON data consists of two objects: {name: 'a', pename: 'b'} and {name: 'c', pename: 'd'}. The response tab shows a 200 OK status with the same JSON data returned.

45

Partitionare de stari intre controlere Spring

• Unidirectional!

```
@Controller
@RequestMapping("/")
public class UnController {

    private int nrDeCereri=0;

    @GetMapping("/")
    @ResponseBody
    public String index(){
        nrDeCereri++;
        return "Hello world!";
    }

    public int getNrDeCereri() {
        return nrDeCereri;
    }
}
```

```
@Controller
@RequestMapping("/alt")
public class AltController {
    private UnController ctrl;
    public AltController(UnController uc){
        this.ctrl=uc;
    }

    @GetMapping("/")
    @ResponseBody
    public String index(){
        return "Numar cereri:" + ctrl.getNrDeCereri();
    }
}
```

The screenshot shows a browser window with the URL localhost:8080/alt. The page content is 'Numar cereri: 2', indicating that the AltController is handling the request and returning the count from the UnController.

Aplicatii Web cu Suport Java, sem.I, 2024-2025

46

46

Partitionare de stari intre controlere Spring

- Bidirectionala!

```
@Controller  
@RequestMapping("/")  
public class UnController {  
  
    private Contor contor;  
  
    private UnController(Contor co) {  
        this.contor = co;  
    }  
  
    @GetMapping("/")  
    @ResponseBody  
    public String index() {  
        contor.inc();  
        return "Hello world!";  
    }  
}
```

```
@Service  
public class Contor {  
    private int contor=0;  
  
    public void inc(){  
        contor++;  
    }  
    public int getContor(){  
        return contor;  
    }  
}
```

```
@Controller  
@RequestMapping("/alt")  
public class AltController {  
    private Contor contor;  
    private AltController(Contor co){  
        this.contor=co;  
    }  
    @GetMapping("/")  
    @ResponseBody  
    public String index(){  
        return "Numar cereri: " + contor.getContor();  
    }  
}
```

Aplicatii Web cu Suport Java, sem.I, 2024-2025

47

47

Cereri REST

Cum sa ignori ca ai de-a face cu JSON!

- Maparea JSON

```
{  
    "nume" : "Dobrica"  
    "prenume" : "Liliana"  
}
```

```
public class Persoana {  
    public String nume;  
    public String prenume;  
}
```

Aplicatii Web cu Suport Java, sem.I, 2024-2025

48

48

Definirea structurilor de date ale modelului

-

Structurile de date pot expune detalii de implementare!

Metode esentiale pentru structurile de date

```
public class Persoana {
    public String nume;
    public String prenume;

    public Persoana(String nume, String prenume){
        this.nume= nume;
        this.prenume= prenume;
    }

    //Constructor fara parametrii pentru mapare
    public Persoana(){
    }

    public static Persoana create(String nume, String prenume){
        Persoana p= new Persoana();
        p.nume= nume;
        p.prenume= prenume;
        return p;
    }

    public int hashCode(){}

    public boolean equals(Object obj){}

    public String toString(){}
}
```

Structurile de date
Nu trebuie sa aiba
algoritmi avansati !

Metoda esentiala
pentru depanare

Aplicatii Web cu Suport Java, sem.I, 2024-2025

49

49

Utilizarea unui Object Mapper JSON

```
public class Persoana {
    public String nume;
    public String prenume;

    public Persoana(String nume, String prenume){
        this.nume= nume;
        this.prenume= prenume;
    }

    //Constructor fara parametrii pentru mapare
    private Persoana(){}
}

public static Persoana create(String nume, String prenume){
    Persoana p= new Persoana();
    p.nume= nume;
    p.prenume= prenume;
    return p;
}

public int hashCode(){}

public boolean equals(Object obj){}

public String toString(){}
}
```

```
ObjectMapper OM=new ObjectMapper();
String json="";
Persoana a=new Persoana("Dobrica", "Liliana");
try {
    json= OM.writeValueAsString(a);
} catch (JsonProcessingException e) {
    throw new RuntimeException(e);
}
System.out.println(json);
```

{"nume":"Dobrica", "prenume": "Liliana"}

```
Persoana b;
try {
    b=OM.readValue(json, Persoana.class);
} catch (JsonProcessingException e) {
    throw new RuntimeException(e);
}
if(a!=b && a.equals(b)){
    System.out.println("equals!");
}
```

equals!
Reimplementati codul din partea dreapta!
•Creati o noua clasa Main2 in proiect
•Adaugati metoda main()
•Copiat codul si executati

Aplicatii Web cu Suport Java, sem.I, 2024-2025

50

50

Spring REST controller

The screenshot shows a Java code editor on the left and a Postman interface on the right.

Java Code (Person.java):

```
public class Persoana {
    public String nume;
    public String prenume;

    public Persoana(String nume, String prenume){
        this.nume= nume;
        this.prenume=prenume;
    }

    //Constructor fara parametrii pentru mapare
    private Persoana(){
    }

    public static Persoana create(String nume, String prenume){
        Persoana p= new Persoana();
        p.nume=nume;
        p.prenume=prenume;
        return p;
    }

    public int hashCode(){}

    public boolean equals(Object obj){}

    public String toString(){}
}
```

Postman Requests:

- GET Request:** URL: `http://localhost:8080/api/personal`. Response status: 200 OK. Body: [empty]
- POST Request:** URL: `http://localhost:8080/api/personal`. Method: POST. Body: `{"nume": "aaa", "prenume": "aaaa"}`. Response status: 200 OK. Body: `[{"id": 1, "nume": "aaa", "prenume": "aaaa"}, {"id": 2, "nume": "aaa", "prenume": "aaaa"}, {"id": 3, "nume": "aaa", "prenume": "aaaa"}, {"id": 4, "nume": "aaa", "prenume": "aaaa"}]`

Aplicatii Web cu Suport Java, sem.I, 2024-2025

51