Академија струковних
студија Шумадија
Одсек Крагујевац

Студијски програм: Информатика
Предмет: Рачунарске мреже

Rent a car

- Пројекат –

Предметни наставник:

Александар Мишковић

Студент:

Никола Тепшић 071/2021,

Крагујевац 2023.

Садржај

Увод

Што се тиче моје теме, израдио сам веб апликацију за потребе рент-а-кар куће. Ова апликација је заснована на RESTful сервисима и има за циљ праћење резервација рентованих возила. Корисници могу да изврше резервацију путем апликације, унесући све потребне информације о времену резервације, типу возила и другим детаљима.

Додатно, ова веб апликација пружа могућност за унос и брисање података из базе. Користећи Постмен, администратори могу лако додавати нове возиле у базу података, унети информације о моделу, годишту, цени и другим релевантним информацијама. Исто тако, могуће је и брисање возила из базе када они више нису доступни за изнајмљивање.

Целокупна апликација је пројектована и развијена у складу са принципима REST архитектуре, што омогућава лаку комуникацију између клијената и сервера преко HTTP протокола. Клијенти могу приступити свим функционалностима апликације преко одговарајућих REST API позива, омогућавајући им ефикасно и удобно коришћење резервационих услуга рент-а-кар куће.

Шта је Spring Framework?

Spring Framework је популаран фрејмворк за развој апликација у Јава програмском језику. Он пружа комплетан скуп алатки и библиотека за развој различитих врста апликација, укључујући веб апликације, веб сервисе, апликације за обраду података и много других.

Spring Framework се истиче по својој лакоћи у употреби и флексибилности. Он укључује инверзију контроле (IoC) и интеграцију са системима за управљање трансакцијама, што олакшава развој апликација. Такође, Spring нуди и модуле за управљање подацима, као што је Spring Data, који олакшава комуникацију са базама података.

Уз то, **Spring Framework** промовише добру архитектуру апликације и подржава примену образаца пројектовања, као што су Модел-Виђет-Контролер (MVC) и Служба-орјентисани архитектурни (SOA) принципи. Ово омогућава да апликација буде добро организована, лако одржива и проширива.

Укратко, **Spring Framework** је моћан алат за развој Јава апликација, који олакшава развој и подржава добру архитектуру. Он је широко коришћен у индустрији и представља популаран избор за програмере који желе да развију ефикасне и модерне апликације.

Апликација

RentCarApplication

Класична **main класа Spring апликације**.

```java
package com.asss.pj;

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

import org.springframework.stereotype.Component;

1 usage
@SpringBootApplication

public class RentaCarApplication {

    no usages
    public static void main(String[] args) {
        SpringApplication.run(RentaCarApplication.class, args);
    }

}
```

Controller

```java
no usages
@RestController

public class RentaCarRestController {
    11 usages
    @Autowired
    private RentaCarServ rentaCarServ;

    no usages
    @GetMapping("/rentacar")
    List<RentaCarEntity> findAll() { return rentaCarServ.findAll(); }


    no usages
    @GetMapping("/carsbypower")
    List<RentaCarEntity> findAllPower() { return rentaCarServ.findAll(Sort.by( ...properties: "Zapremina_Motora")); }

    no usages
    @PostMapping("/addcar")
    RentaCarEntity addCar(@RequestBody RentaCarEntity rentaCarEntity) { return rentaCarServ.save(rentaCarEntity); }

    no usages
    @PutMapping("/editcar")
    RentaCarEntity editCar(@RequestBody RentaCarEntity rentaCarEntity) { return rentaCarServ.save(rentaCarEntity); }

    no usages
    @DeleteMapping("/removecar/id")
    String removeCar(@PathVariable int id) {
        RentaCarEntity rentaCarEntity = rentaCarServ.findById(id);
        if (rentaCarServ != null) {
            rentaCarServ.deleteById(id);
            LoggerHelper.DostupnihVozilaSaBenzinom(rentaCarServ.findAll());
            LoggerHelper.DostupnihVozilaSaDizelom(rentaCarServ.findAll());
            LoggerHelper.DostupnihVozilaNaGas(rentaCarServ.findAll());
            LoggerHelper.strujniPogon(rentaCarServ.findAll());
            return "Automobil (" + id + ") je obrisana";
        } else return "Automobil (" + id + ") nije pronadjena";
    }
}
```

Овај код представља класу RentaCarRestController која је означена анотацијом @RestController, што означава да је ова класа задужена за обраду HTTP захтева и враћање резултата у формату подобном за веб.

У оквиру ове класе, имамо инјекцију зависности (@Autowired) за RentaCarServ сервис. Ова инстанца сервиса користи се за извршавање различитих операција над RentaCarEntity ентитетима.

Уз помоћ анотација @GetMapping, @PostMapping, @PutMapping и @DeleteMapping, дефинисане су различите HTTP руте и методе за приступ ресурсима.

findAll() метод, приступајући путањи /rentacar и користећи GET метод, враћа листу свих RentaCarEntity ентитета.

findAllPower() метод, приступајући путањи /carsbypower и користећи GET метод, враћа листу свих RentaCarEntity ентитета сортирану по пољу Zapremina_Motora.

addCar() метод, приступајући путањи /addcar и користећи POST метод, додава нови RentaCarEntity у базу података. Подаци о новом возилу прослеђују се у телу захтева (@RequestBody анотација).

editCar() метод, приступајући путањи /editcar и користећи PUT метод, врши измену постојећег RentaCarEntity ентитета у бази података. Подаци о изменама прослеђују се у телу захтева.

removeCar() метод, приступајући путањи /removecar/{id} и користећи DELETE метод, уклања RentaCarEntity ентитет из базе података на основу задатог идентификатора (id). Такође, извршавају се одређене операције помоћу LoggerHelper класе.

Entity

```java
32 usages
@Entity
@Table(name = "auto")
public class RentaCarEntity {


    4 usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENT

    @Column(name = "ID")
    private int ID;

    4 usages
    @Column(name = "Marka_Automobila")
    private String Marka_Automobila;

    4 usages
    @Column(name = "Model")
    private String Model;

    4 usages
    @Column(name = "Kilometraza")
    private int Kilometraza;

    4 usages
    @Column(name = "zapremina_Motora")
    private int zapremina_Motora;

    4 usages
    @Column(name = "Tip_Goriva")
    @Enumerated(EnumType.STRING)
    private com.asss.pj.util.Tip_Goriva Tip_Goriva;
    4 usages
    @Column(name = "Cena")
    private Integer cena;
```

```java
    //GETTERI I SETTERI

    no usages
    public int getID() { return ID; }

    no usages
    public void setID(int ID) { this.ID = ID; }

    no usages
    public String getMarka_Automobila() { return Marka_Automobila; }

    no usages
    public void setMarka_Automobila(String marka_Automobila) { this.Marka_Automobila = marka_Automobila; }

    no usages
    public String getModel() { return Model; }

    no usages
    public void setModel(String model) { this.Model = model; }

    no usages
    public int getKilometraza() { return Kilometraza; }

    no usages
    public void setKilometraza(int kilometraza) { this.Kilometraza = kilometraza; }

    no usages
    public int getZapremina_Motora() { return zapremina_Motora; }

    no usages
    public void setZapremina_Motora(int zapremina_Motora) { this.zapremina_Motora = zapremina_Motora; }

    4 usages
    public com.asss.pj.util.Tip_Goriva getTip_Goriva() { return Tip_Goriva; }

    no usages
    public void setTip_Goriva(com.asss.pj.util.Tip_Goriva tip_Goriva) { this.Tip_Goriva = tip_Goriva; }

    no usages
    public Integer getCena() { return cena; }

    no usages
    public void setCena(Integer cena) { this.cena = cena; }
```

```java
    public RentaCarEntity(int ID, String marka_Automobila, String model, int kilometraza, int zapremina_Motora, com.asss.pj.util.Tip_Goriva tip_Goriva, Integer cena) {
        this.ID = ID;
        this.Marka_Automobila = marka_Automobila;
        this.Model = model;
        this.Kilometraza = kilometraza;
        this.zapremina_Motora = zapremina_Motora;
        this.Tip_Goriva = tip_Goriva;
        this.cena = cena;
    }
```

Овај код представља класу **RentaCarEntity** која представља ентитет у бази података за аутомобиле у систему рент-а-кар.

Ова класа је означена анотацијом **@Entity**, што означава да представља ентитет у бази података. Анотација **@Table(name = "auto")** означава да је ова класа мапирана на табелу "auto" у бази података.

У овој класи имамо следећа поља:

- **ID** - означава идентификациони број аутомобила и означено је анотацијом **@Id** и **@GeneratedValue** која омогућава аутоматску генерацију вредности за ово поље.
- **Marka_Automobila** - означава марку аутомобила и означено је анотацијом **@Column(name = "Marka_Automobila")**.
- **Model** - означава модел аутомобила и означено је анотацијом **@Column(name = "Model")**.
- **Kilometraza** - означава пређену километражу аутомобила и означено је анотацијом **@Column(name = "Kilometraza")**.
- **zapremina_Motora** - означава запремину мотора аутомобила и означено је анотацијом **@Column(name = "zapremina_Motora")**.
- **Tip_Goriva** - означава тип горива аутомобила и означено је анотацијом **@Column(name = "Tip_Goriva")** и **@Enumerated(EnumType.STRING)** која указује да је ово поле енумерацијски тип.
- **cena** - означава цену аутомобила и означено је анотацијом **@Column(name = "Cena")**.

Класа има конструкторе, гетере и сетере за сва поља, што омогућава приступ и постављање вредности поља.

Такође, преписана је метода **toString()** која враћа стринг репрезентацију објекта **RentaCarEntity**, укључујући сва поља класе.

Repository

```java
package com.asss.pj.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.asss.pj.entity.RentaCarEntity;
2 usages
@Repository
public interface RentaCarRepository extends JpaRepository<RentaCarEntity, Integer> {
}
```

Овај код представља интерфејс **RentaCarRepository** који је одговоран за комуникацију са базом података за ентитет **RentaCarEntity**.

Овај интерфејс екстендује **JpaRepository<RentaCarEntity, Integer>**, што омогућава коришћење стандардних CRUD (Create, Read, Update, Delete) операција над ентитетима.

Анотација **@Repository** означава да је ова класа репозиторијум, што је специфичан тип компоненте у Spring Framework-у.

Овај интерфејс нуди следеће функционалности:

- Наслеђује методе за рад са базом података као што су **findAll()**, **findById()**, **save()**, **deleteById()** итд.
- Проширује се могућностима специфичним за **RentaCarEntity** ентитет, као што је претрага и сортирање података на основу различитих критеријума.

Овај интерфејс омогућава апстракцију слоја за приступ подацима и смањује потребу за писањем SQL упита ручно, јер Spring Data JPA аутоматски генерише SQL упите на основу дефинисаних метода интерфејса.

Сервиси

RentaCarServ

```
package com.asss.pj.service;

import ...

7 usages  1 implementation
public interface RentaCarServ {
    5 usages  1 implementation
    List<RentaCarEntity> findAll();

    2 usages  1 implementation
    RentaCarEntity save(RentaCarEntity rentaCarEntity);

    1 usage  1 implementation
    String deleteById(int id);

    1 usage  1 implementation
    RentaCarEntity findById(int id);

    1 usage  1 implementation
    List<RentaCarEntity> findAll(Sort Zapremina_Motora);
}
```

Овај код представља интерфејс **RentaCarServ** који дефинише сервисне операције за рад са ентитетом **RentaCarEntity**.

Овај интерфејс има следеће методе:

- **findAll()**: Враћа све рентацар аутомобиле из базе података.
- **save(RentaCarEntity rentaCarEntity)**: Чува нови или ажурира постојећи рентацар аутомобил у бази података.
- **deleteById(int id)**: Брише рентацар аутомобил из базе података на основу идентификационог броја.
- **findById(int id)**: Проналази и враћа рентацар аутомобил из базе података на основу идентификационог броја.
- **findAll(Sort Zapremina_Motora)**: Враћа све рентацар аутомобиле из базе података сортиране по пољу **Zapremina_Motora** у одређеном редоследу.

Овај интерфејс дефинише операције које се користе за управљање рентацар аутомобилима, као што су претрага, додавање, ажурирање и

брисање. Имплементација ових метода би требало да се налази у одговарајућој класи сервиса која имплементира овај интерфејс.

RentaCarServImpl

```java
package com.asss.pj.service;
import ...
no usages
@Service
public class RentaCarServImpl implements RentaCarServ {
    6 usages
    @Autowired
    private RentaCarRepository rentaCarRepository;
    1 usage
    @Override
    public RentaCarEntity findById(int ID) {
        return rentaCarRepository.findById(ID).orElse( other: null);
    }
    5 usages
    @Override
    public List<RentaCarEntity> findAll() {
        return rentaCarRepository.findAll();
    }
    2 usages
    @Override
    public RentaCarEntity save(RentaCarEntity rentaCarEntity) {
        return rentaCarRepository.save(rentaCarEntity);
    }
    1 usage
    @Override
    public String deleteById(int ID) {
        RentaCarEntity rentaCarEntity = rentaCarRepository.findById(ID).orElse( other: null);
        if (rentaCarEntity != null) {
            rentaCarRepository.deleteById(ID);
            return "Automobil (" + ID + ") je obrisan!";
        } else {
            return "Automobil (" + ID + ") nije pronadjen";
        }
    }
    1 usage
    @Override
    public List<RentaCarEntity> findAll(Sort Zapremina_Motora) {
        return rentaCarRepository.findAll(Sort.by(Sort.Direction.DESC,  ...properties: "Zapremina_Motora"));
    }
}
```

Овај код представља имплементацију интерфејса **RentaCarServ**. Класа **RentaCarServImpl** је анотирана са **@Service**, што означава да је она сервисни компонент у Spring Framework-у.

Ова имплементација користи **RentaCarRepository** за комуникацију са базом података.

Он имплементира све методе дефинисане у интерфејсу **RentaCarServ** и пружа следеће функционалности:

- **findById(int ID)**: Проналази и враћа рентацар аутомобил на основу задатог идентификационог броја.
- **findAll()**: Враћа све рентацар аутомобиле из базе података.
- **save(RentaCarEntity rentaCarEntity)**: Чува нови или ажурира постојећи рентацар аутомобил у бази података.
- **deleteById(int ID)**: Брише рентацар аутомобил из базе података на основу задатог идентификационог броја.
- **findAll(Sort Zapremina_Motora)**: Враћа све рентацар аутомобиле из базе података сортиране по полју **Zapremina_Motora** у опадајућем редоследу.

Ова имплементација користи методе из **RentaCarRepository** да би обавила одговарајуће операције са базом података.

Util

Logger

```
no usages
@AfterReturning("getMethod()")
public void gMethod() {
    try (FileWriter fileWriter = new FileWriter(LoggerHelper.getFileName(), append: true)) {
        fileWriter.write( str: "[" + LocalDate.now() + "]" + " [" + LocalTime.now().truncatedTo(ChronoUnit.SECONDS) + "]" + "---Log: GET metoda koriscena" + "\n");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
no usages
@AfterReturning("getPower()")
public void gpMethod() {
    try (FileWriter fileWriter = new FileWriter(LoggerHelper.getFileName(), append: true)) {
        fileWriter.write( str: "[" + LocalDate.now() + "]" + " [" + LocalTime.now().truncatedTo(ChronoUnit.SECONDS) + "]" + "---Log: GET po ceni sortirani" + "\n");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
no usages
@AfterReturning("putMethod()")
public void puMethod() {
    try (FileWriter fileWriter = new FileWriter(LoggerHelper.getFileName(), append: true)) {
        fileWriter.write( str: "[" + LocalDate.now() + "]" + " [" + LocalTime.now().truncatedTo(ChronoUnit.SECONDS) + "]" + "---Log: PUT metoda koriscena" + "\n");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
no usages
@AfterReturning("deleteMethod()")
public void delMethod() {
    try (FileWriter fileWriter = new FileWriter(LoggerHelper.getFileName(), append: true)) {
        fileWriter.write( str: "[" + LocalDate.now() + "]" + " [" + LocalTime.now().truncatedTo(ChronoUnit.SECONDS) + "]" + "---Log: DELETE metoda koriscena" + "\n");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
no usages
@AfterReturning("postMethod()")
public void poMethod() {
    try (FileWriter fileWriter = new FileWriter(LoggerHelper.getFileName(), append: true)) {
        fileWriter.write( str: "[" + LocalDate.now() + "]" + " [" + LocalTime.now().truncatedTo(ChronoUnit.SECONDS) + "]" + "---Log: POST metoda koriscena" + "\n");
    } catch (IOException e) {
```

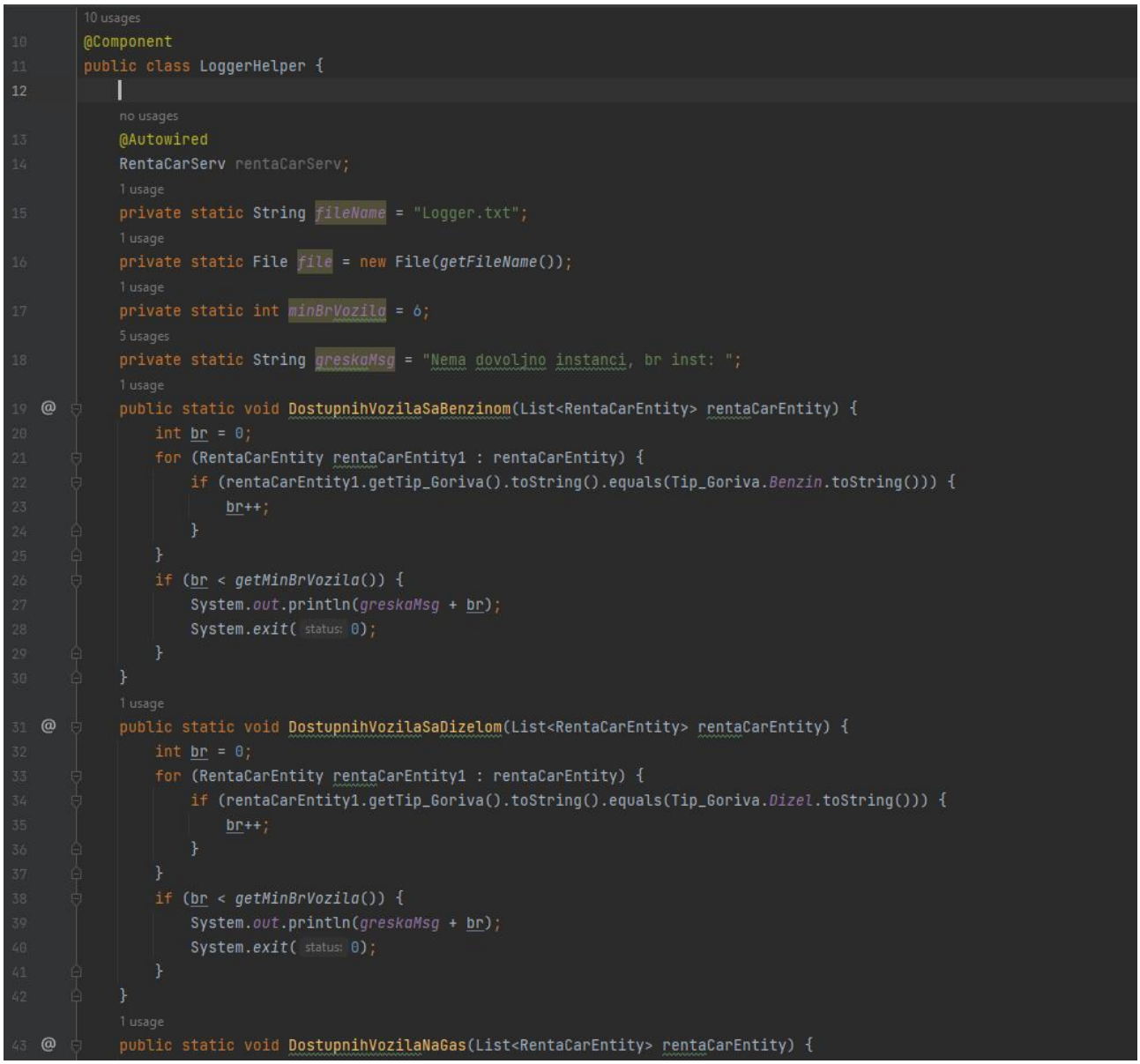
Овај код представља класу **Logger** која служи за записивање лог информација о извршеним операцијама у систему. Класа је анотирана са **@Aspect**, што означава да је она аспект у Spring Framework-у.

Он имплементира следеће функционалности:

- Дефинише различите **@Pointcut** анотације које одређују тачке извршавања за различите методе у **RentaCarRestController** контролеру.
- Користи се **@AfterReturning** анотација за обраду после успешног повратка из метода одређене тачке извршавања.
- У свакој обрађивачкој методи, записује лог информације у датотеку користећи **FileWriter**.
- Лог информације укључују датум, време и тип извршене операције.

Ова класа се користи за праћење и документовање различитих акција које се извршавају у систему и може помоћи у откривању проблема, анализи и прилагођавању система.

LoggerHelper



```
10 usages
@Component
public class LoggerHelper {

    no usages
    @Autowired
    RentaCarServ rentaCarServ;

    1 usage
    private static String fileName = "Logger.txt";

    1 usage
    private static File file = new File(getFileName());

    1 usage
    private static int minBrVozila = 6;

    5 usages
    private static String greskaMsg = "Nema dovoljno instanci, br inst: ";

    1 usage
    public static void DostupnihVozilaSaBenzinom(List<RentaCarEntity> rentaCarEntity) {
        int br = 0;
        for (RentaCarEntity rentaCarEntity1 : rentaCarEntity) {
            if (rentaCarEntity1.getTip_Goriva().toString().equals(Tip_Goriva.Benzin.toString())) {
                br++;
            }
        }
        if (br < getMinBrVozila()) {
            System.out.println(greskaMsg + br);
            System.exit( status: 0);
        }
    }

    1 usage
    public static void DostupnihVozilaSaDizelom(List<RentaCarEntity> rentaCarEntity) {
        int br = 0;
        for (RentaCarEntity rentaCarEntity1 : rentaCarEntity) {
            if (rentaCarEntity1.getTip_Goriva().toString().equals(Tip_Goriva.Dizel.toString())) {
                br++;
            }
        }
        if (br < getMinBrVozila()) {
            System.out.println(greskaMsg + br);
            System.exit( status: 0);
        }
    }

    1 usage
    public static void DostupnihVozilaNaGas(List<RentaCarEntity> rentaCarEntity) {
```

Овај код представља класу **LoggerHelper** која служи за помоћне функционалности у оквиру логера. Класа је анотирана са **@Component**, што означава да се третира као компонента у Spring Framework-у.

Он имплементира следеће функционалности:

- Дефинише статичне методе **DostupnihVozilaSaBenzinom**, **DostupnihVozilaSaDizelom**, **DostupnihVozilaNaGas** и **strujniPogon** које проверавају количину доступних возила са одређеним типом горива.
- Уколико је број доступних возила мањи од **minBrVozila**, исписује се порука о грешци и програм се завршава.

- Подаци о датотеци и променљиве потребне за проверу количине доступних возила су декларисане као статичке променљиве.
- Статички метод **getFileName** враћа име датотеке за логирање.

Ова класа служи за проверу услова и акција које се извршавају у систему, као што је провера количине доступних возила са одређеним типом горива.

Enum – Tip_Goriva

```
1  package com.asss.pj.util;
2
   8 usages
3  public enum Tip_Goriva {
       3 usages
4      Benzin,Dizel,Gas,Struja;
5  }
6
```

Овај код представља енумерацију **Tip_Goriva**. Енумерације су специјални типови у Јава програмирању који омогућавају дефинисање константи које представљају одређене вредности. У овом случају, енумерација дефинише различите типове горива који могу бити додељени аутомобилима.

Конкретно, **Tip_Goriva** има четири константе: **Benzin**, **Dizel**, **Gas** и **Struja**. Ове константе представљају могуће типове горива који могу бити додељени објектима класе **RentaCarEntity** (како је дефинисано у коду који сте претходно представили). Оваква дефиниција енумерације омогућава да се ограничи избор вредности за поле **Tip_Goriva** на тачно дефинисане вредности.

Закључак

Укратко, ова апликација за управљање ренталним аутомобилом представља комплетан систем за преглед, додавање, измену и брисање информација о аутомобилима. Коришћени су модули и компоненте како би се организовало и раздвојило одговорности између различитих делова система.

Архитектура апликације следи добру праксу раздвајања слојева, где су контролери одговорни за обраду HTTP захтева, сервиси за имплементацију бизнисне логике, репозиторијуми за комуникацију са базом података, и ентитети који представљају моделе података.