Spring / Hibernate

DUOMENŲ ĮTERPIMAS, ATNAUJINIMAS, IŠTRYNIMAS

Užduotis

Sukurkime naują projektą kuris bus skirtas baigiamųjų darbų talpinimui ir peržiūrai

Jį sudarys tokios klasės ir DB:

Students

```
id
name
surname
gender
birth_year
```

Works

```
id
student_id
supervisor_id    - int
name             - string
year             - int
type                    - tiny int: 0, 1
(bachelor, master)
keywords                - string
```

Supervisor

```
Id
name
surname
title            - string
phd              - tiny int (yes/no)
```

Užduotis

Sukurkime **entities** klases. Sukurkime DAO **interfeisus** ir jų realizacijas.

StudentDAO

StudentDAOImpl

WorkDAO

WorkDAOImpl

SupervisorDAO

SupervisorDAOImpl

Interfeisuose turi būti šie metodai:

Paimti visą sąrašą

Pridėti paduotą įrašą

Paimti vieną įrašą pagal ID

Atnaujinti paduotą įrašą

Ištrinti paduotą įrašą

Užduotis

Sukurkimve visus metodus reikalingus visų studentų sąrašo atvaizdavimui.

Atvaizduokime visus studentus.

DAO metodo getStudents() įgyvendinimas

```
public List<Task> getStudents(){
 //Sesijos paėmimas
  Session session= sessionFactory.getCurrentSession();

 //Užklausos visiem įrašams paimti generavimas
 Query<Student> query=session.createQuery("from Student", Student.class);

 //Įrašų paėmimas
  List<Student> students=query.getResultList();

//Įrašų gražinimas
  return students;
}
```

Sukursime kontrolerį StudentController

Sukurkime kontrolerį StudentController.

Kontroleryje mums bus reikalingas StudentDAO objektas. Jį galime gauti pasinaudodami Spring Dependency injection kontroleryje nurodę:

@Autowired

private StudentDAO studentDAO;

Jis bus prijungtas dėl to, kad StudentDAO interfeiso implementacijoje nurodėme @Repository anotaciją.

Objektas studentDAO bus klasės studentDAOImpl.

Duomenų perdavimas į JSP failą

Metodas skirtas duomenų perdavimui galėtų atrodyti taip:

```
public String(Model model){
  //Paimamos visos užduotys
  List<Student> students=studentDAO.getStudents();

  //Įdedame duomenis į modelį
  model.addAttribute("students",students);

  //Atvaizduojame tasks.jsp failą
  return "students";
}
```

Sukurkime JSP failą atvaizdavimui

Atvaizduojant panaudokime JSTL tagus:

```
<c:forEach var="students" items="${students}">

</c:forEach>
```

Puslapio atvaizdavimas

Pagal nutylėjimą, užėjus į projektą kraunamas tas failas kuris yra nurodytas web.xml faile (arba turintis bazinį route).

Norėdami iš JSP failo padaryti nukreipimą, galime parašyti:

<% response.sendRedirect("students/list"); %>

@GetMapping ir @PostMapping

Mes jau naudojome anotaciją:

@RequestMapping – ji priima ir GET ir POST metodu siunčiamus duomenis ir atvaizduoja juos

Taip pat galime naudoti ir

@GetMapping – apdoros tik užklausas siųstas GET metodu

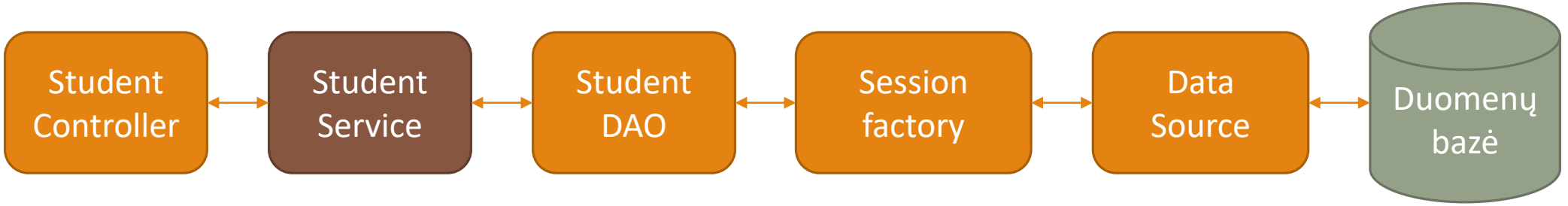
@PostMapping –apdoros tik užklausas siųstas POST metodu

@GetMapping("/processForm") atliktų tą patį ką ir

@RequestMapping(path="/processForm", method=RequestMethod.GET)

Service layer

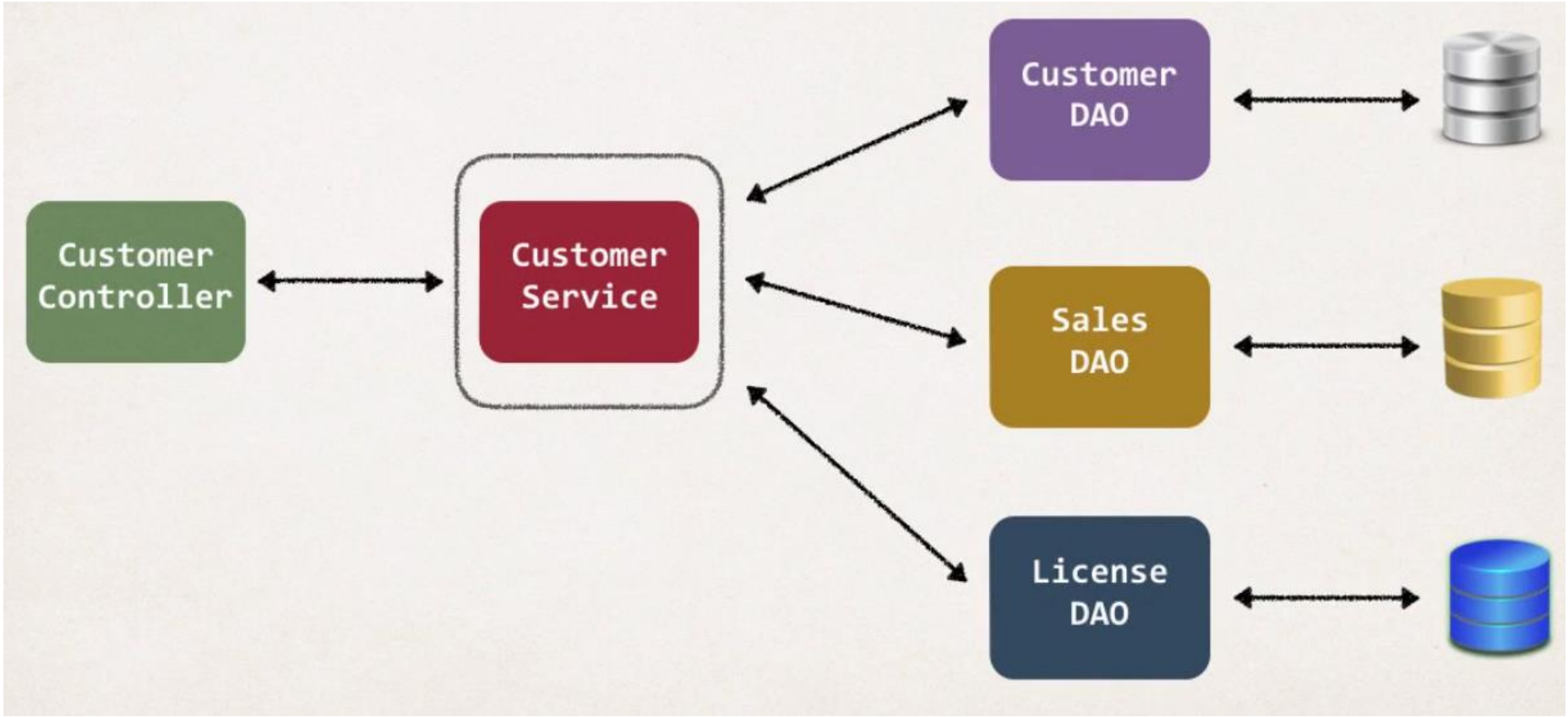
Prie mūsų kuriamos programos pridėkime papildomą sluoksnį: Service Layer



Service Layer reiktų naudoti todėl, kad:

- padės įgyvendinti sistemoje Service Facade projektavimo šabloną,
- Tarpinis sluoksnis kuriame įgyvendinama visa veiklos logika
- Integruoja duomenis iš daug šaltinių (DAO/repositorijų)

Service Layer integruoja daug DAO



@Service anotacija

@Service anotacijų pagalba sukuriami Service‘s

Spring automatiškai priregistruos Service‘s klases, todėl bus galima vykdyti DI

Services įgyendinamos panašiai kaip DAO klasės:

- Sukuriamas Service interfeisas su visais metodais
- Sukuriama Service interfeisą įgyvendinanti klasė
- Atliekamas DI su visais DAO

Sukurkime service interfeisą

Sukurkime naują paketą .services, jame sukurkime interfeisą StudentService

Sukurkime vieną metodą:

```
public interface StudentService {
   public List<Student> getStudents();
}
```

Sukurkime TasksService interfeisą įgyvendinančią klasę

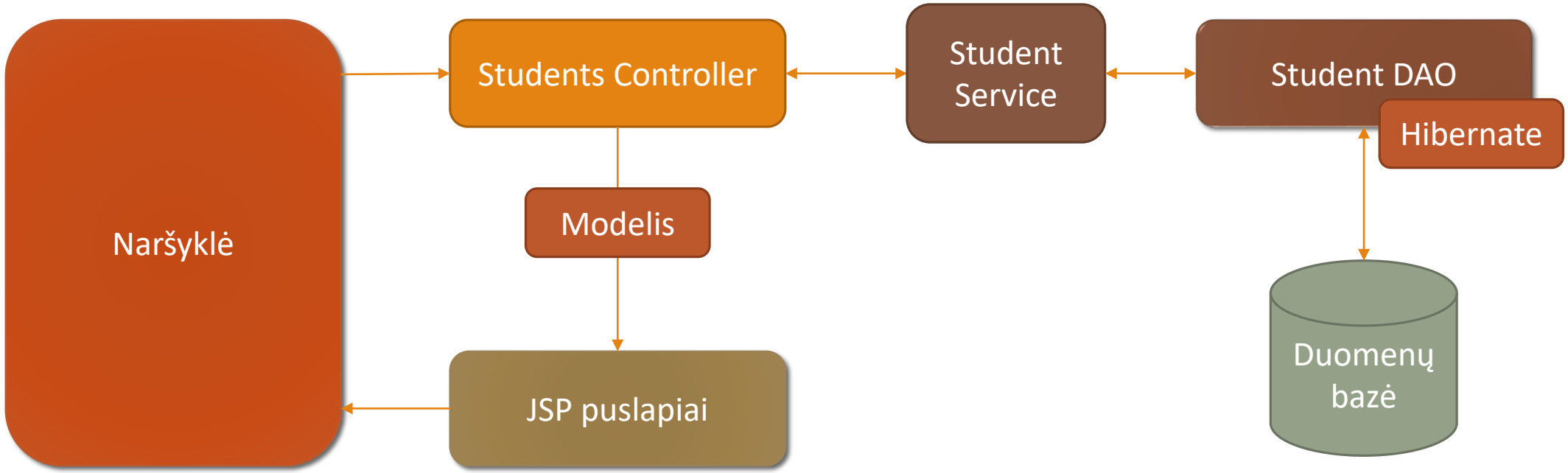
Sukurkime klasę:

```
@Service
public class StudentsServiceImpl implements StudentsService {
   @Autowired
   private StudentDAO studentDAO;

  @Transactional
  public List<Student> getStudents(){
  // ….
  }
}
```

Kadangi @Transactional perkėlėme į Service layer, ją turėtumėme pašalinti iš DAO klasės

Bendra sistemos schema



Kontrolerio modifikacija

Modifikuokime kontrolerį taip, kad jame nebenaudotumėme DAO, o naudotumėmės tik Service.

Jį taip pat galime prisidėti tokiu būdu:

@Autowired

Private StudentService studentService;

Užduotis. Naujų įrašų pridėjimas

Pamėginkime suprogramuoti naujos užduoties pridėjimą, tam mums reikės:

- Atnaujinti JSP (pridedant linką į užduoties pridėjimą)
- Sukurti naują HTML formą užduoties pridėjimui
- Atnaujinti:
  - Controller‘i pridedant HTML formos rodymą ir duomenų paėmimą
  - Service – pridedant metodus naujos užduoties pridėjimą
  - DAO – sukurti metodą naujos užduoties išsaugojimui

Užduotis

Pridėkime naują mygtuką virš užduočių sąrašo: „Pridėti naują studentą“

Padarykime to mygtuko nuorodą: /student/addStudent

Jame turėsime sukurti HTML formą su įvedimo laukais:

Vardas, Pavardė, Amžius, Lytis ir mygtukas submit

Užduotis: atnaujinkime JSP ir kontrolerį

Kontroleryje sukurkime naują route /addStudent (GET tipo) kuris rodys šią formą.

Šiame route sukurkime naują objektą: Student ir jį perduokime į JSP

JSP faile sukurkime formą kurioje bus atvaizduotas šis objektas:

- <form:form action="saveTask" modelAttribute="task" method="POST">
- </form:form>

Drop-down sąrašai

Spring MVC karkase <select> tagą atitiktų:

```
<form:select path="parametras">
  <form:option value="reiksme" label="antraštė">
  <form:option value="reiksme" label="antraštė">
  <form:option value="reiksme" label="antraštė">
</form:select>
```

Užduotis

Pamėginkime padaryti jog pridedant naują studentą jo lytis būtų pasirenkama iš sąrašo:

Vyras / Moteris

Select parametrų padavimas iš klasių

```
<form:select path="parametras">
    <form:options items="${kintamasis}" itemValue="value" itemLabel="label" />
</form:select>
```

Užduotis

Sukurkime metų sąrašą (HashMap tipo) ir jį padarykime kaip selectą metų pasirinkimui.

Redirect’as iš kontrolerio

Norėdami nukreipti vartotoją iš kontrolerio (pavyzdžiui pridėjus duomenis jei norėtumėme vartotoją nukreipti į duomenų sąrašą), tuomet, mūsų kontroleris turėtų gražinti tekstą:

redirect:/url_i_kur_nukreipti

Pavyzdžiui

```
public String saveTask(@ModelAttribute("task") Task task){


   return "redirect:/tasks/list";

}
```

StudentService atnaujinimas

StudentService sukurkime metodą skirtą užduoties išsaugojimui

Service interface sukurkime metodą skirtą užduoties išsaugojimui

- public void saveStudent (Student student);

Service implementacijoje sukurkime metodo saveStudent (Student student) realizaciją.

- Metodas turėtų tiesiog persiųsti duomenis į DAO metodą saveStudent(Student student)

StudentDAO atnaujinimas

StudentDAO sukurkime metodą skirtą užduoties išsaugojimui

StudentDAO interface sukurkime metodą skirtą užduoties išsaugojimui

- public void saveStudent(Student student);

DAO implementacijoje sukurkime metodo saveStudent(Student student) realizaciją.

- Metodas turėtų pasiimti hibernate sessiją ir išsaugoti duomenis:
  - Session session = sessionFactory.getCurrentSession();
  - session.save(student);

Užduotis. Įrašo atnaujinimas

Pamėginkime suprogramuoti studento atnaujinimą (kad eitų redaguoti studentą),tam mums reikės:

- Atnaujinti JSP (pridedant linką į studento atnaujinimą)
- Sukurti HTML formą užduoties koregavimui
- Atnaujinti:
  - Controller‘i pridedant HTML formos rodymą su duomenimis
  - Service – pridedant metodus studento išsaugojimui
  - DAO – sukurti metodą studento išsaugojimui

Užduotis. Įrašo ištrynimas

Pamėginkime suprogramuoti užduoties ištrynimą (kad eitų ištrinti studentą),tam mums reikės:

- Atnaujinti JSP (pridedant linką į studento ištrynimą)
- Atnaujinti:
  - Service – pridedant metodus studento ištrynimui
  - DAO – sukurti metodą studento ištrynimui
  - Kontroleryje sukurti route