



Ondokuz Mayıs Üniversitesi, Web Programlama Labı

Ad - Soyad: Alper Karaca

Öğrenci Numarası: 21060619

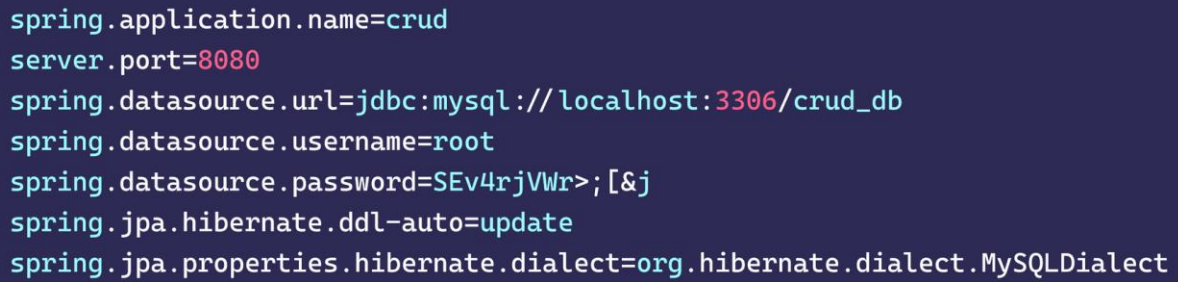
Ödev Adı: Föy - 5

Özet Bilgi

Bu f6y ierisinde Java Springboot, Hibernate ve Thymeleaf kullanarak CRUD iřlemlerini gerekleřtireceėiz.

Kaynak Kodları

1. Application.properties



```
spring.application.name=crud
server.port=8080
spring.datasource.url=jdbc:mysql://localhost:3306/crud_db
spring.datasource.username=root
spring.datasource.password=SEv4rjVWr>[&j
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
```

Bu dosya Springboot uygulamamızda kullanılacak bilgilerin tutulduğu bir özellik dosyasıdır.

İçerisinde Web sunucusunun portu 8080 olarak ayarlanmıştır ve veritabanına bağlanmak için gerekli olan bilgileri tutmaktadır.

2. MainModel.java

```
package com.foy5.crud.model;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity(name = "employee")
public class MainModel {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    public Integer empID;

    @Column(nullable = false)
    public String empName;

    @Column(nullable = false)
    public String empEmail;

    @Column(nullable = false, length = 14)
    public String empPhone;

    @Column(nullable = false)
    public String empDepartment;

    @Column(nullable = false)
    public String empRole;

    @Column(nullable = false)
    public String empPicture = "https://thispersondoesnotexist.com/";
    // .....
}
```

Bu dosya içerisinde veritabanında oluşturulacak tablolar ve veri tipleri, özellikleri gibi verilerin tutulduğu model dosyasıdır.

ID için otomatik üretme özelliği, telefon numarasına uzunluk sınırı ve fotoğraf içinse bir varsayılan değer ayarlanmıştır.

Kodun devamında tüm setter – getter fonksiyonları ve constructorları mevcuttur.

3. MainController.java

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

    @Autowired
    private MainRepository repository;
    // Bu fonksiyon site üzerinden rastgele bir kişi resmi alır
    private String generatePersonImage() {
        return "https://thispersondoesnotexist.com/";
    }

    // Read Operation
    @GetMapping
    public String listEmployees(Model model) {
        List<MainModel> employees = repository.findAll();

        employees = employees.stream()
            .map(emp -> {
                if (emp.getEmpPicture() == null || emp.getEmpPicture().isEmpty()) {
                    emp.setEmpPicture(generatePersonImage());
                }
                return emp;
            })
            .collect(Collectors.toList());

        model.addAttribute("employees", employees);
        return "index";
    }
}
```

Kodun üst kısmında gerekli olan bütün import ve paket isimlendirmesi işlemleri yapılmıştır.

@Controller notasyonu ile bu dosyanın bir controller dosyası olduğu belirtilmiştir ve tüm istekler ana dizin (/) üzerinden yapılmakta olduğu için @RequestMapping(/) notasyonu kullanılmıştır.

@AutoWired notasyonu sayesinde bağımlılık enjeksiyonu otomatik biçimde sağlanmıştır.

generatePersonImage() fonksiyonu thispersondoesntexist.com üzerinden rastgele bir fotoğraf urlsi alır ve döndürür.

@GetMapping notasyonu bu URL üzerinde yapılacak GET isteklerine izin verir.

listEmployees() fonksiyonu view tarafına gönderebilmek için bir Model objesi alır.

Daha sonrasında repository dosyası ve hibernate sayesinde tablodaki tüm veriler çekilir ve akış (stream) haline getirilir. Bu akış üzerinde resmi olmayan kullanıcılara resim atanır ve listeye çevrilip model üzerine eklenir. Bu işlemler sonucunda gösterilecek olan sayfa ismi olarak ise “index” gösterilir.

```

// Kullanıcı verilerini çekmek için Endpoint ve en sonunda
//çekilen veriler ilgili Fragment üzerinde döndürülür
@GetMapping("/list")
public String listEmployeesFragment(Model model) {
    List<MainModel> employees = repository.findAll();

    // Eğer resim yoksa rastgele bir görsel ata
    employees = employees.stream()
        .map(emp -> {
            if (emp.getEmpPicture() == null || emp.getEmpPicture().isEmpty()) {
                emp.setEmpPicture(generatePersonImage());
            }
            return emp;
        })
        .collect(Collectors.toList());

    model.addAttribute("employees", employees);
    return "fragments/employee-list";
}

//Kullanıcı düzenleme işlemi eğer kullanıcı resim belirtmediyse rastgele bir resim atar
@GetMapping("/edit/{id}")
public String editEmployee(@PathVariable int id, Model model) {
    Optional<MainModel> employee = repository.findById(id);

    if (employee.isPresent()) {
        MainModel emp = employee.get();
        if (emp.getEmpPicture() == null || emp.getEmpPicture().isEmpty()) {
            emp.setEmpPicture(generatePersonImage());
        }
        model.addAttribute("employee", emp);
    }

    return "fragments/employee-form";
}

```

/list endpointine gelen istekler de listEmployees fonksiyonuyla aynı işlemi yapar ancak buradaki veri index yerine employee-form fragmentine gönderilir.

```
// Create Operation
@PostMapping("/save")
public String saveEmployee(@ModelAttribute MainModel employee) {
    // Eğer resim yoksa rastgele bir görsel ata
    if (employee.getEmpPicture() == null || employee.getEmpPicture().isEmpty()) {
        employee.setEmpPicture(generatePersonImage());
    }

    System.out.println(employee);

    repository.save(employee);
    return "redirect:/";
}

// Delete Operation
@PostMapping("/delete/{id}")
public String deleteEmployee(@PathVariable int id) {
    repository.deleteById(id);
    return "redirect:/";
}
}
```

Kaydetme işlemi bir POST isteği olduğu için `@PostMapping(/save)` notasyonu ile bu endpointe gelen veriler veritabanına kaydedilir.

Burada verilerin doğruluğunun kontrolü Thymeleaf ile yapıldığı için bu örnek uygulamada backend tarafında veri doğrulaması işlemi yapılmamıştır. En sonunda `save()` fonksiyonu ile obje veritabanına kaydedilir ve index dosyasının olduğu sayfa `"redirect:/"` ile yenilenir.

Silme işlemi Thymeleaf ile POST isteği üzerinden yapıldığı için bir tane daha `@PostMapping` notasyonu kullandık ve endpoint sonunda Path değişkeni olarak ID'yi gönderdik.

`deleteEmployee()` fonksiyonu ise Hibernate içerisinde tanımlı olan `deleteById` fonksiyonu ile ilgili ID'ye sahip objeyi silecektir ve sonrasında kullanıcı ana sayfaya yönlendirilecektir.

4. index.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  <head>
    <meta charset="UTF-8" />
    <link th:href="@{/main.css}" rel="stylesheet" />
    <title>Çalışan Yönetim Sistemi</title>
  </head>
  <body class="bg-gray-100 p-8">
    <div class="container mx-auto bg-white shadow-lg rounded-lg p-6">
      <h1 class="text-3xl font-bold mb-6 text-center text-gray-800">
        Çalışan Yönetim Sistemi
      </h1>

      <div class="mb-6">
        <form
          th:action="@{/save}"
          method="post"
          class="grid grid-cols-1 md:grid-cols-2 gap-4"
        >
          <input type="hidden" name="empID" th:value="${employee?.empID}" />

          <div>
            <label class="block text-gray-700 mb-2">Ad Soyad</label>
            <input
              type="text"
              name="empName"
              th:value="${employee?.empName}"
              required
              class="w-full px-3 py-2 border rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500"
            />
          </div>

          <div>
            <label class="block text-gray-700 mb-2">Email</label>
            <input
              type="email"
              name="empEmail"
              th:value="${employee?.empEmail}"
              required
              class="w-full px-3 py-2 border rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500"
            />
          </div>

          <div>
            <label class="block text-gray-700 mb-2">Telefon</label>
            <input
              type="tel"
              name="empPhone"
              th:value="${employee?.empPhone}"
              required
              maxlength="14"
              class="w-full px-3 py-2 border rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500"
            />
          </div>
        </form>
      </div>
    </div>
  </body>
</html>
```



```
<div>
  <label class="block text-gray-700 mb-2">Departman</label>
  <select
    name="empDepartment"
    class="w-full px-3 py-2 border rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500"
    required
  >
    <option value="">Departman Seçin</option>
    <option
      value="IT"
      th:selected="{employee?.empDepartment === 'IT'}"
    >
      Bilgi Teknolojileri
    </option>
    <option
      value="HR"
      th:selected="{employee?.empDepartment === 'HR'}"
    >
      İnsan Kaynakları
    </option>
    <option
      value="Finance"
      th:selected="{employee?.empDepartment === 'Finance'}"
    >
      Finans
    </option>
    <option
      value="Marketing"
      th:selected="{employee?.empDepartment === 'Marketing'}"
    >
      Pazarlama
    </option>
  </select>
</div>

<div>
  <label class="block text-gray-700 mb-2">Rol</label>
  <select
    name="empRole"
    class="w-full px-3 py-2 border rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500"
    required
  >
    <option value="">Rol Seçin</option>
    <option
      value="Developer"
      th:selected="{employee?.empRole === 'Developer'}"
    >
      Geliştirici
    </option>
    <option
      value="Manager"
      th:selected="{employee?.empRole === 'Manager'}"
    >
      Yönetici
    </option>
    <option
      value="Analyst"
      th:selected="{employee?.empRole === 'Analyst'}"
    >
      Analist
    </option>
    <option
      value="Support"
      th:selected="{employee?.empRole === 'Support'}"
    >
      Destek
    </option>
  </select>
</div>
```

```

<div>
  <label class="block text-gray-700 mb-2">Profil Resmi</label>
  <input
    type="text"
    name="empPicture"
    th:value="{employee?.empPicture}"
    class="w-full px-3 py-2 border rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500"
    placeholder="Resim URL'si"
  />
</div>

<div class="col-span-full">
  <button
    type="submit"
    class="w-full bg-blue-500 text-white py-2 rounded-lg hover:bg-blue-600 transition duration-300"
  >
    Kaydet
  </button>
</div>
</form>
</div>

<div
  id="employeeList"
  class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4"
>
  <div
    th:each="emp : {employees}"
    class="bg-gray-50 p-4 rounded-lg shadow-md flex flex-col items-center"
  >
    
    <h3
      class="font-bold text-lg text-center"
      th:text="{emp.empName}"
    ></h3>
    <p class="text-gray-600 text-center" th:text="{emp.empEmail}"></p>
    <p
      class="text-gray-500 text-center"
      th:text="{emp.empDepartment + ' - ' + emp.empRole}"
    ></p>
    <div class="mt-4 flex space-x-2">
      <a
        th:href="{@/edit/{id}(id={emp.empID})}"
        class="bg-yellow-500 text-white px-3 py-1 rounded hover:bg-yellow-600"
      >
        Düzenle
      </a>
      <form
        th:action="{@/delete/{id}(id={emp.empID})}"
        method="post"
        onsubmit="return confirm('Bu çalışanı silmek istediğinizden emin misiniz?');"
      >
        <button
          type="submit"
          class="bg-red-500 text-white px-3 py-1 rounded hover:bg-red-600"
        >
          Sil
        </button>
      </form>
    </div>
  </div>
</div>
</div>
</div>
</body>
</html>

```

Öncelikle sayfayı tasarlarken frontend kısmında TailwindCSS ve Thymeleaf kullandım.

SpringBoot ile TailwindCSS nasıl kullanılır konusu [bu bağlantıda](#) mevcuttur.

Adımları takip ettikten sonra elimde oluşan main.css dosyasını Thymeleaf ile ekledikten sonra içerisinde İsim, Email, Telefon Numarası, Rol, Departman ve Profil Fotoğrafı için gerekli veri girme alanlarını barındıran ve /save endpointine istek gönderen bir form ve bir kaydetme butonu ekliyorum. Bu kayıt formunun alt tarafında her satırda 3 kişi görülecek şekilde bir çalışanlar tablosu oluşturdum.

Bu tablodaki çalışan objesinin her birinin sağ tarafında Sil ve Düzenle butonları bulunmaktadır.

Çalışanların fotoğrafları, isimleri, mail adresleri ve Departman – Rol bilgileri gösterilmektedir.

Bu bilgilerin tutulduğu dosyalar fragment olarak tutulduğu için aslında HTML dosyasına bir nevi HTML enjeksiyonu yapıyoruz diyebiliriz. Bu sayede birden fazla HTML dosyası tek bir sayfa içerisinde renderlenmektedir.

5. employee-list.html

```
37 <!DOCTYPE html>
38 <html xmlns:th="http://www.thymeleaf.org">
39   <head>
40     <meta charset="UTF-8" />
41     <link th:href="@{/main.css}" rel="stylesheet" />
42     <title>Çalışan Yönetim Sistemi</title>
43   </head>
44   <body>
45     <div
46       th:fragment="employeeList"
47       class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4"
48     >
49       <div
50         th:each="emp : ${employees}"
51         class="bg-gray-50 p-4 rounded-lg shadow-md flex flex-col items-center"
52       >
53         
59         <h3 class="font-bold text-lg text-center" th:text="${emp.empName}"></h3>
60         <p class="text-gray-600 text-center" th:text="${emp.empEmail}"></p>
61         <p
62           class="text-gray-500 text-center"
63           th:text="${emp.empDepartment + ' - ' + emp.empRole}"
64         ></p>
65         <div class="mt-4 flex space-x-2">
66           <button
67             class="bg-yellow-500 text-white px-3 py-1 rounded hover:bg-yellow-600"
68           >
69             Düzenle
70           </button>
71           <button
72             th:post="${'/delete/' + emp.empID}"
73             th:confirm="Bu çalışanı silmek istediğinize emin misiniz? Bu işlem geri alınamaz."
74             class="bg-red-500 text-white px-3 py-1 rounded hover:bg-red-600"
75           >
76             Sil
77           </button>
78         </div>
79       </div>
80     <div
81       th:if="${#lists.isEmpty(employees)}"
82       class="col-span-full text-center text-gray-500 p-6"
83     >
84       Henüz hiç çalışan eklenmemiş.
85     </div>
86   </div>
87 </body>
88 </html>
```

Kullanıcıların listelendiği fragments dosyası. Th:each fonksiyonu ile her bir employee objesi için bir bölüm oluşturulur ve bilgileri ile doldurulur.

6. employee-form.html

[illegible]

Bu dosya içerisinde kullanıcı bilgilerini güncellemek için gerekli olan veri alanları mevcuttur. Ayrıca geri çıkma ve kaydetme butonu da mevcuttur.

Ekran Görüntüleri

Çalışan Yönetim Sistemi

Ad Soyad: Test Kullanıcı

Email: test@test.com

Telefon: 55555555555555

Departman: Pazarlama

Rol: Analist

Profil Resmi: https://this-person-does-not-exist.com/img/avatar-gene58d22bb4111811c703e6ff0e5786416.jpg

Kaydet

Alper Karaca
alpeerkaraca@gmail.com
IT - Developer

Düzenle **Sil**

Çalışan Yönetim Sistemi

Ad Soyad:

Email:

Telefon:

Departman: Departman Seçin

Rol: Rol Seçin

Profil Resmi: Resim URL'si

Kaydet

Alper Karaca
alpeerkaraca@gmail.com
IT - Developer

Düzenle **Sil**

Test Kullanıcı
test@test.com
Marketing - Analyst

Düzenle **Sil**

GitHubalperkaraca's ProfileInstagramCarbonGmail-21Gmail-ls

http://localhost:8080/edit/16

Çalışan Bilgilerini Düzenle

Ad Soyad

Test Kullanıcı Düzenleniyor !

Email

test@test.com

Telefon

55555555555555

Departman

Pazarlama

Rol

Analist

Profil Resmi

https://this-person-does-not-exist.com/img/avatar-gene58d22

GüncelleKapat

GitHubalperkaraca's ProfileInstagramCarbonGmail-21Gmail-ls

http://localhost:8080

Çalışan Yönetim Sistemi

Ad Soyad

Email

Telefon

Departman

Departman Seçin


Rol

Rol Seçin

Profil Resmi


Resim URL'si

Kaydet



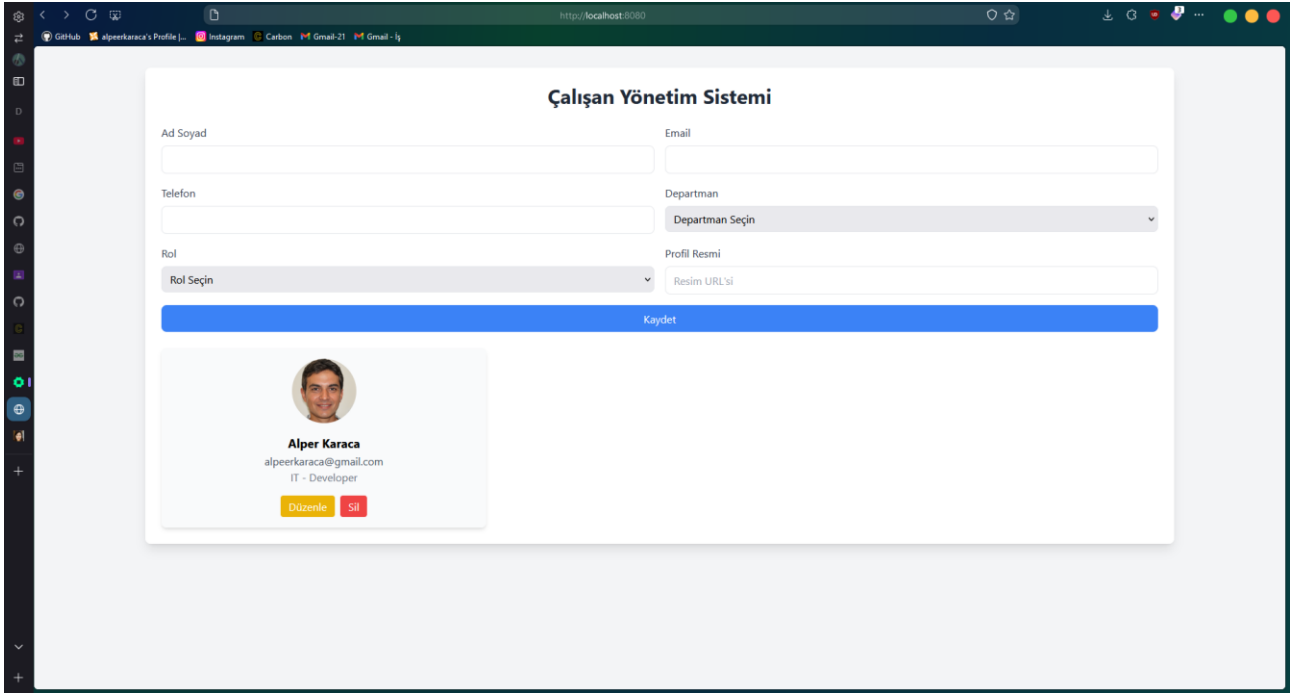
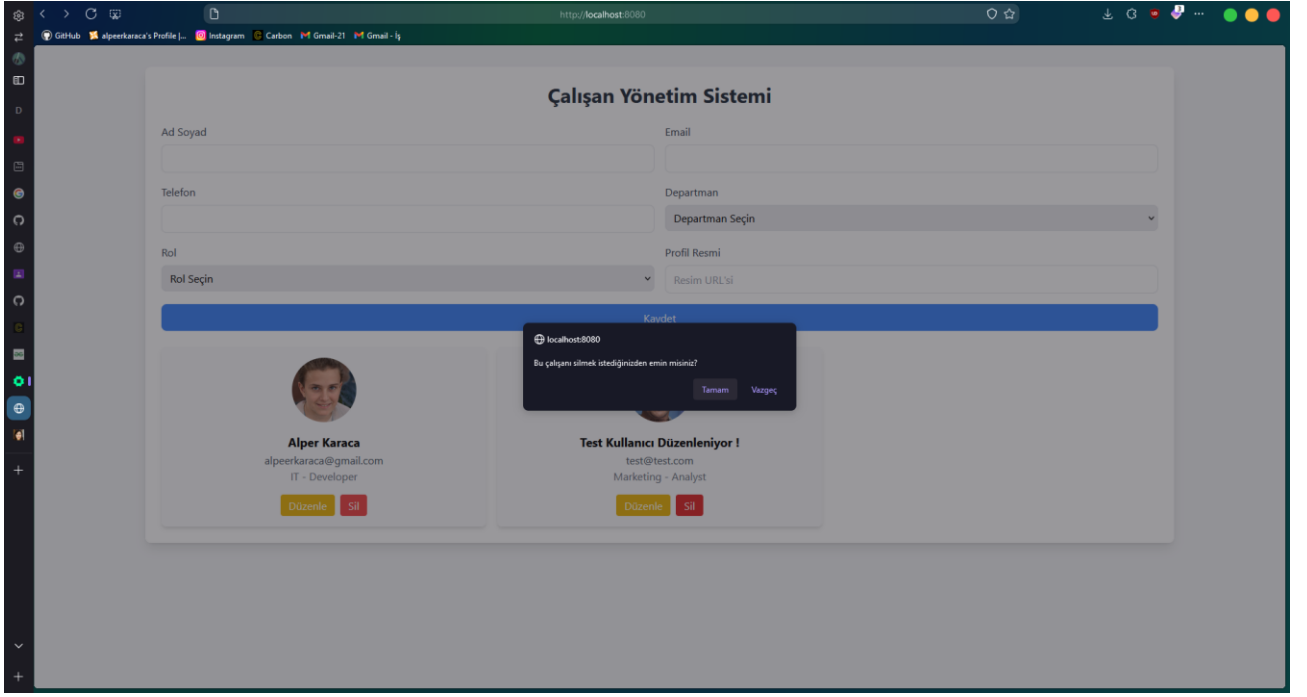
Alper Karaca
alperkaraca@gmail.com
IT - Developer

DüzenleSil



Test Kullanıcı Düzenleniyor !
test@test.com
Marketing - Analyst

DüzenleSil



Çalışan Yönetim Sistemi

Ad Soyad

Email

Telefon

Departman

Departman Seçin


Rol

Rol Seçin

Profil Resmi

Resim URL'si


Kaydet



Alper Karaca
alpeerkaraca@gmail.com
IT - Developer

Düzenle


Sil



Test 1
asfasf@asfasf.com
HR - Developer

Düzenle


Sil



Test 2
ashsdksjdg@asfdsaf.com
HR - Manager

Düzenle

Sil



Test - 3
test3@test.com
IT - Analyst

Düzenle

Sil