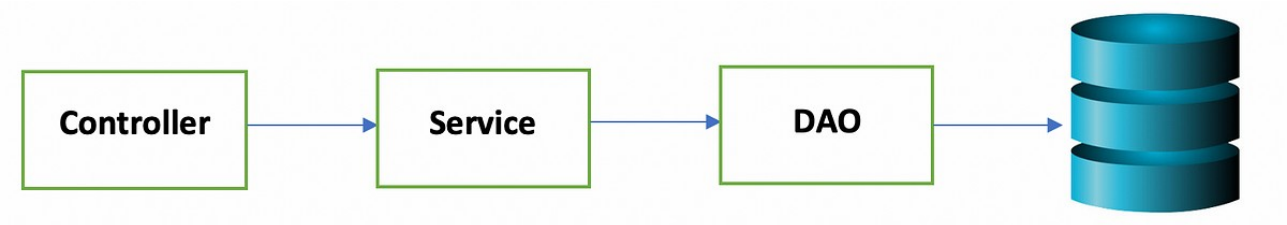
Spring Data JPA

DAO ları oluşturmak için bir pattern olan spring data JPA yı kullandım.Bu sayede birden fazla DAO kullanıldığı zaman her biri için implemente etmeme gerek kalmadı.CRUD methodlarını bu sayede elde edebildik.

```
package com.spring.springboot.caseStudy.dao;

import ...

3 usages
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {
```

Service Layer



Service layer employee controller ile DAO arasında yer alacak.Birden fazla kaynaktan gelen verileri entegre etmek için kullanılacak.Ama bizim için şuan tek bir DAO var.

```
package com.spring.springboot.caseStudy.service;

import com.spring.springboot.caseStudy.entity.Employee;

import java.util.List;

4 usages  1 implementation
public interface EmployeeService {

    1 usage  1 implementation
    List<Employee> findAll();

    2 usages  1 implementation
    Employee findById(int theId);

    1 usage  1 implementation
    Employee save(Employee theEmployee);

    1 usage  1 implementation
    void deleteById(int theId);

}
```

Service Interface:

CRUD metholarını interface'imizde tanımladık.

Service Implementation :

En üstte @Service ek acıklaması spring bileşen taraması sayesinde bu bileşeni otomatik kayıt edecek. EmployeeDAO constructorı inject edilir.Daha sonra çağrıları EmployeeDAO devrediyoruz.Controllerın doğrudan DAO kullanması yerine service i kullanılması sağlandı.

```java
@Service
public class EmployeeServiceImpl implements EmployeeService {

    5 usages
    private EmployeeRepository employeeRepository;

    @Autowired
    public EmployeeServiceImpl(EmployeeRepository theEmployeeRepository) { employeeRepository = theEmployeeRepository; }

    1 usage
    @Override
    public List<Employee> findAll() { return employeeRepository.findAll(); }

    2 usages
    @Override
    public Employee findById(int theId) {
        Optional<Employee> result = employeeRepository.findById(theId);

        Employee theEmployee = null;

        if (result.isPresent()) {
            theEmployee = result.get();
        }
        else {

            throw new RuntimeException("Did not find employee id - " + theId);
        }

        return theEmployee;
    }

    1 usage
    @Override
    public Employee save(Employee theEmployee) { return employeeRepository.save(theEmployee); }

    1 usage
    @Override
    public void deleteById(int theId) { employeeRepository.deleteById(theId); }
}
```

Controller

Service için constructor injection yapıldı.

```
private EmployeeService employeeService;
public EmployeeController(EmployeeService theEmployeeService)
{
   employeeService=theEmployeeService;
}
```

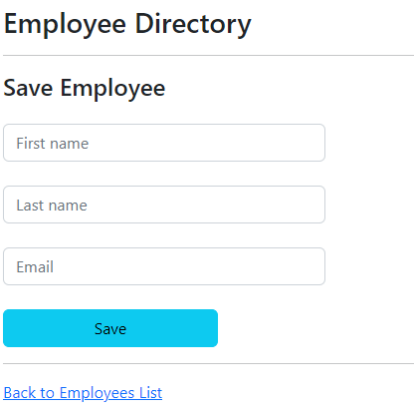
List ile databasedeki tüm veriler getirildi.Databaseden gelen veriler html dosyasını içinde yazılan yere aktarıldı.

```
@GetMapping("/list")
public String listEmployees(Model theModel)
{
   List<Employee> theEmployees =employeeService.findAll();
   theModel.addAttribute("employees",theEmployees);

   return "employees/list-employees";
}
```

First Name	Last Name	Email
refik orkun	arslan	refikorkunarslan@gmail.com

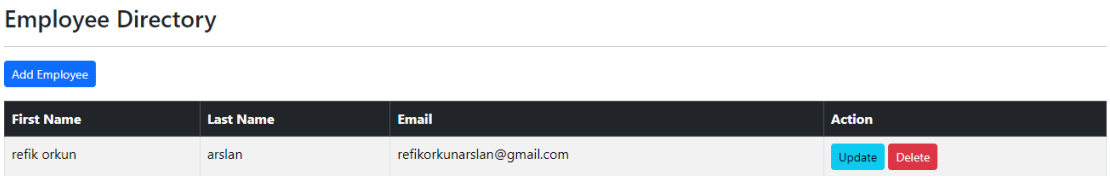
Employee ekleme için önce bir employee oluşturuyoruz .Model içindeki rolunu atadıktan sonra employee-form olan html sayfasına yönlendiriyoruz.

```
@GetMapping("/AddEmp")
public String AddEmp(Model theModel) {


   Employee theEmployee = new Employee();

   theModel.addAttribute("employee", theEmployee);

   return "employees/employee-form";
}
```



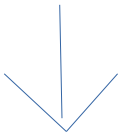
Daha sonra bu forma yazılan bilgileri databasemize kaydediyoruz.Return olarak da verileri listelediğimiz yere dönüyoruz.

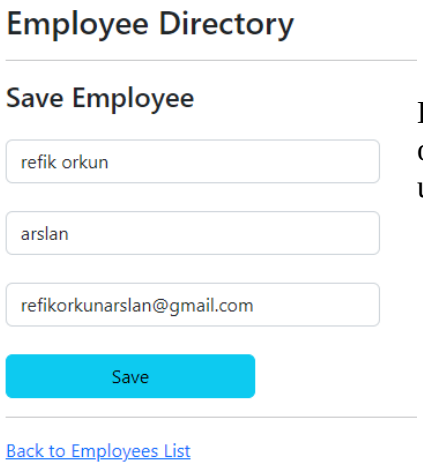
```
@PostMapping("/save")
public String saveEmployee(@ModelAttribute("employee") Employee theEmployee)
{
   employeeService.save(theEmployee);
   return "redirect:/employees/list";
}
```



Update de ise değiştirilmek istenen verinin id'sini alıp ekleme için kullanılan forma yönlendirdim.

```
@GetMapping("/UpdateEmp")
public String showFormForUpdate(@RequestParam("employeeId") int theId,
                                Model theModel) {

   Employee theEmployee = employeeService.findById(theId);
   theModel.addAttribute("employee", theEmployee);
   return "employees/employee-form";
}
```

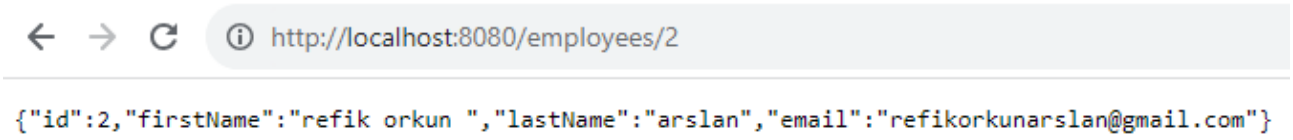




Direkt olarak update butonuna basılan veriler formda otamatik olarak dolu geldi.Daha sonra değiştirilip save a basıldığımda update işlemi gerçekleşir.

Employee id ile direkt olarak o employee yi döndürme şansı da mevcut.

```
@GetMapping("/{employeeId}")
@ResponseBody
public Employee getEmployee(@PathVariable int employeeId) {

    Employee theEmployee = employeeService.findById(employeeId);

    if (theEmployee == null) {
        throw new RuntimeException("Employee id not found - " + employeeId);
    }

    return theEmployee;
}
```



{"id":2,"firstName":"refik orkun ","lastName":"arslan","email":"refikorkunarslan@gmail.com"}