

Full Stack Developer Bitirme Projesi

ERKAN UZUNCA

02.09.2023

Java ve React ile TodoList Uygulaması

BACKEND

- TaskDto (Data Transfer Object) Class
- TaskEntity Class
- IRepository
- ITaskService
- TaskServiceImpl
- ITaskApi
- TaskApiImpl

FRONTEND

- TaskCreate
- TaskList
- TaskUpdate
- TaskView
- TaskApi

TaskDto & TaskEntity

TASKDTO

Veri aktarımı için kullandığımız sınıf
3 değişkene sahiptir

```
private Long taskId;  
private String taskName;  
private boolean taskCompleted;
```

TASKENTITY

Veritabanı tablomuzu temsil ediyor

```
@Log4j2  
@Entity  
@Table(name = "tasks")  
public class TaskEntity extends AuditingAwareBaseEntity implements Serializable {  
  
    private static final Long serialVersionUID = 1L;  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    @Column(name = "tasks_id", unique = true, nullable = false, insertable = true, updatable = false)  
    private Long taskId;  
  
    @Column(name = "tasks_name")  
    private String taskName;  
  
    @Column(name = "tasks_completed")  
    private boolean taskCompleted;
```

IRepository & ITaskServices

IREPOSITORY

Veritabanı işlemleri için arabirim

```
@Repository
public interface ITaskRepository extends CrudRepository<TaskEntity, Long> {

    no usages 1 Erkan
    Optional<TaskEntity> findByTaskName(String taskName);
}
```

ITASKSERVICES

İş mantığının soyutlanmış hali

CRUD işlemlerinin gerçekleştiği yer

```
public interface ITaskServices<D, E> {

    // Model Mapper
    2 usages 1 implementation 1 Erkan
    public D entityToDto(E e);

    2 usages 1 implementation 1 Erkan
    public E dtoToEntity(D d);
}
```

```
// C R U D
// CREATE
1 usage 1 implementation 1 Erkan
public D taskServiceCreate(D d);

// LIST
1 usage 1 implementation 1 Erkan
public List<D> taskServiceList();

// FIND BY
3 usages 1 implementation 1 Erkan
public D taskServiceFindById(Long id);

// UPDATE
1 usage 1 implementation 1 Erkan
public D taskServiceUpdate(Long id, D d);

// DELETE
1 usage 1 implementation 1 Erkan
public D taskServiceDeleteById(Long id);
}
```

TaskServicesImpl

TASKSERVICESIMPL

ITaskServices'in uygulandığı yer
Veritabanı işlemleri ve iş mantığının
entegrasyonu burada sağlanır.
Api ile veritabanı işlemleri arasındaki
köprüyü oluşturur.

```
public TaskDto taskServiceCreate(TaskDto taskDto) {  
    if (taskDto != null) {  
        TaskEntity taskEntity = dtoToEntity(taskDto);  
        iTaskRepository.save(taskEntity);  
        taskDto.setTaskId(taskEntity.getTaskId());  
    } else {  
        throw new NullPointerException("TaskDto is null");  
    }  
    return taskDto;  
}
```

ITaskApi & TaskApiImpl

ITASKAPI

Api endpointleri'ni tanımlar

```
// C R U D
// CREATE
no usages 1 implementation 1 Erkan
ResponseEntity<?> taskApiCreate(D d);
```

TASKAPIIMPL

ITaskApi burada uygulanır

Endpoint işlemleri için iş mantığı

```
@RestController
@CrossOrigin(origins = "http://localhost:3000")
@RequestMapping("/task/api/v1")
public class TaskApiImpl implements ITaskApi<TaskDto> {

    private final ITaskServices iTaskServices;

    //http://localhost:4444/task/api/v1/create

    1 Erkan
    @Override
    @PostMapping("/create")
    public ResponseEntity<?> taskApiCreate(@Valid @RequestBody TaskDto taskDto) {
        return ResponseEntity.ok(iTaskServices.taskServiceCreate(taskDto));
    }
}
```

FrontEnd

TASKAPI

TaskApi.js dosyamız oluşturuldu.

Gerekli CRUD metodları oluşturuldu.

```
class TaskApi {  
  
  // CREATE  
  // http://localhost:4444/category/api/v1/create  
  // @PostMapping("/create")</CategoryDto>  
  taskApiCreate(taskDto) {  
    return axios.post(`${TASK_URL}/create`, taskDto)  
  };  
}
```

TASKLIST.JSX

Değişken tanımlamaları yapıldı.

Gerekli fonksiyonlar oluşturuldu.


```
const setUpdateTask = (data) => {  
  const { id, taskName, systemDate, completed } = data;  
  localStorage.setItem("task_update_id", id);  
  localStorage.setItem("task_update_task_name", taskName);  
  localStorage.setItem("task_update_task_date", systemDate);  
  localStorage.setItem("task_update_completed", completed);  
};
```

FrontEnd

EKRAN ÇIKTISI

Son olarak karşımızda programın ekran çıktısı bulunuyor.

ToDoInput

 New Todo















Add new task

ToDoList

All

Done

Todo

Görev1 Görev1 Görev12	<input checked="" type="checkbox"/>		
Görev1 Görev1	<input type="checkbox"/>		
Görev1	<input checked="" type="checkbox"/>		
Görev3	<input checked="" type="checkbox"/>		
Task1	<input type="checkbox"/>		
görev33333	<input checked="" type="checkbox"/>		
Sample Task	<input type="checkbox"/>		

Delete All Tasks

Delete Completed Tasks