





# BaseDto & BaseEntity

## BaseDto

Veri aktarımı için kullandığımız sınıf  
3 değişkene sahiptir

```
@MappedSuperclass
@Data
public abstract class BaseDto {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private LocalDateTime createdAt = LocalDateTime.now();
    @UpdateTimestamp
    private LocalDateTime updatedAt;
}
```

## BaseEntity

Ortak özellikler için olan veritabanı  
tablomuzu temsil ediyor

```
@MappedSuperclass
@Data
public abstract class BaseEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private LocalDateTime createdAt = LocalDateTime.now();
    @UpdateTimestamp
    private LocalDateTime updatedAt;
}
```

# SaveTodoDto&TodoDto& Todo

## SaveTodoDto

Todo'ları kaydetmek için kullanırız.

## TodoDto

Todo'ları listelemek için kullanırız.

## Todo

Todo'larımız için veritabanı tablomuzu temsil eder.

```
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Data
public class SaveTodoDto {

    @Lob
    @NotBlank
    @Size(min = 3)
    private String content;

}
```

```
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Data
public class TodoDto extends BaseDto{

    private String content;

    private Boolean isDone;

}
```

```
@Data
@Entity
@NoArgsConstructor
@AllArgsConstructor
@Builder(toBuilder = true)
@Table(name = "todos")
public class Todo extends BaseEntity{

    @Lob
    private String content;

    @Builder.Default
    private Boolean isDone = false;

}
```

# TodoRepository

TodoRepository, bir JPA (Java Persistence API) repository arayüzünün bir örneğidir ve Todo nesnelerini veritabanıyla etkileşimde bulunmak için kullanılır. Bu arayüzün temel amacı, belirli işlemleri gerçekleştirmek için Todo nesneleri üzerinde veritabanı sorgularını tanımlamak ve bu sorguları kullanarak veritabanı işlemlerini gerçekleştirmektir.

```
public interface TodoRepository extends JpaRepository<Todo, Long> {  
  
    1 usage  🧑 hakanozdmr  
    List<Todo> getTodosByIsDoneTrue();  
  
    1 usage  🧑 hakanozdmr  
    List<Todo> getTodosByIsDoneFalse();  
  
    1 usage  🧑 hakanozdmr  
    void deleteTodosByIsDoneTrue();  
}
```

# TodoService

TodoService sınıfı, uygulamamızda Todo nesneleri üzerinde çeşitli işlemleri gerçekleştirmek için kullanılır.

```
@RequiredArgsConstructor
@Service
public class TodoService {

    private final TodoRepository todoRepository;

    1 usage  🧑 hakanozdmr
    public List<Todo> getAllTodos() { return todoRepository.findAll(); }

    1 usage  🧑 hakanozdmr
    @Transactional
    public List<Todo> getAllTodosIsDoneTrue() { return todoRepository.getTodosByIsDoneTrue(); }

    1 usage  🧑 hakanozdmr
    @Transactional
    public List<Todo> getAllTodosIsDoneFalse() { return todoRepository.getTodosByIsDoneFalse(); }

    1 usage  🧑 hakanozdmr
    public Todo addTodo(SaveTodoDto saveTodoDto){
        return todoRepository.save(Todo.builder().content(saveTodoDto.getContent()).build());
    }

    💡 @Transactional
    > public void deleteTodoIsDoneTrue() { todoRepository.deleteTodosByIsDoneTrue(); }

    1 usage  🧑 hakanozdmr
    > public void deleteTodosAll() { todoRepository.deleteAll(); }

    1 usage  🧑 hakanozdmr
    > public void deleteTodoById(Long id) { todoRepository.deleteById(id); }

    1 usage  🧑 hakanozdmr
    @Transactional
    public Todo updateTodoContent(Long id, String content){
        Todo t = todoRepository.getReferenceById(id);
        t.setContent(content);
        return todoRepository.save(t);
    }

    1 usage  new *
    public void updateTodoIsDone(Long id){
        Todo todo = todoRepository.findById(id).orElse( other: null);
        if (todo != null) {
            todo.setIsDone(!todo.getIsDone());
            todoRepository.save(todo);
        }
    }

}
```

# OpenAPIBean

OpenAPIBean sınıfı,  
Spring Boot  
uygulamamızda Swagger  
OpenAPI belgesini  
yapılandırmak için  
kullanılır.

```
@Configuration
public class OpenAPIBean {

    @Bean
    public OpenAPI customOpenAPI(@Value("Todo API") String description,
                                  @Value("1.0") String version) {

        return new OpenAPI()
            .info(new Info().title("Todo API")
                .version(version)
                .description(description)
                .license(new License().name("Todo API Licence")));
    }
}
```



# TodoController

TodoController sınıfı, Spring Boot uygulamamızın RESTful API'sini tanımlar ve HTTP isteklerine cevap verir. TodoService sınıfını kullanarak Todo nesneleri üzerinde çeşitli işlemleri gerçekleştiren bir API sağlar.

```
@RestController
@RequiredArgsConstructor
@RequestMapping("/v1/todo")
@CrossOrigin(origins = "http://localhost:3000")
public class TodoController {

    private final TodoService todoService;

    // hakanozdmr
    @GetMapping()
    public ResponseEntity<List<Todo>> getAllTodos() { return ResponseEntity.ok(todoService.getAllTodos()); }

    // hakanozdmr
    @GetMapping("/getAllTodosIsDoneTrue")
    public ResponseEntity<List<Todo>> getAllTodosIsDoneTrue(){
        return ResponseEntity.ok(todoService.getAllTodosIsDoneTrue());
    }

    // hakanozdmr
    @GetMapping("/getAllTodosIsDoneFalse")
    public ResponseEntity<List<Todo>> getAllTodosIsDoneFalse(){
        return ResponseEntity.ok(todoService.getAllTodosIsDoneFalse());
    }

    // hakanozdmr *
    @PostMapping
    public ResponseEntity<Todo> addTodo(@RequestBody SaveTodoDto saveTodoDto){
        return ResponseEntity
            .status(HttpStatus.CREATED)
            .body(todoService.addTodo(saveTodoDto));
    }
}
```



# TodoController

```
@DeleteMapping(value = "delete/deleteTodoIsDoneTrue")
public ResponseEntity<Void> deleteTodoIsDoneTrue(){
    todoService.deleteTodoIsDoneTrue();
    return ResponseEntity.noContent().build();
}

// hakanozdmr
@DeleteMapping(value = "delete")
public ResponseEntity<Void> deleteTodosAll(){
    todoService.deleteTodosAll();
    return ResponseEntity.noContent().build();
}

// hakanozdmr
@DeleteMapping(value = "delete/{id}")
public ResponseEntity<Void> deleteTodoById(@PathVariable Long id){
    todoService.deleteTodoById(id);
    return ResponseEntity.noContent().build();
}

// hakanozdmr
@PutMapping(value = "update/{id}/updateContent/")
public ResponseEntity<Todo> updateTodoContent(@PathVariable Long id, @RequestParam String content){
    return ResponseEntity.ok(todoService.updateTodoContent(id, content)) ;
}

// hakanozdmr *
@PutMapping(value = "update/updateTodoIsDone/{id}")
public ResponseEntity<Todo> updateTodoIsDone(@PathVariable Long id){
    todoService.updateTodoIsDone(id);
    return ResponseEntity.noContent().build();
}
}
```

# FrontEnd

## TodoService

TodoService sınıfı, React uygulamamızın ön yüzünden HTTP istekleri göndererek Spring Boot uygulamasının RESTful API'sini kullanır.

```
import axios from "axios";
const API_URL = 'http://localhost:8080/v1/todo';

export default class TodoService {
  getAllTodos(){
    return axios.get(API_URL)
  }
  getAllTodosIsDoneTrue(){
    return axios.get(API_URL+"/getAllTodosIsDoneTrue")
  }
  getAllTodosIsDoneFalse(){
    return axios.get(API_URL+"/getAllTodosIsDoneFalse")
  }
  addTodo(todo){
    return axios.post(API_URL,todo)
  }
  deleteTodosIsDoneTrue(){
    return axios.delete(API_URL+"/delete/deleteTodoIsDoneTrue")
  }
  deleteTodosAll(){
    return axios.delete(API_URL+"/delete")
  }
  deleteTodoById(id){
    return axios.delete(API_URL+`/delete/${id}`)
  }
  updateTodoContent(id,content){
    return axios.put(API_URL+`/update/${id}/updateContent/?content=${content}`)
  }
  updateTodoIsDone(id){
    return axios.put(API_URL+`/update/updateTodoIsDone/${id}`)
  }
}
```

# App.js& Todo.jsx

## App.js

**App.js, React uygulamamızın ana bileşenidir ve uygulamanın temel yapısını oluşturur.**

## Todo.jsx

Todo.jsx, React uygulamanızın ana görüntüsünü oluşturan bir bileşendir. TodoInput ve TodoList bileşenlerini içerir. Bu alt bileşenler, kullanıcının yeni görevler eklemesini ve mevcut görevleri listelemesini sağlar.

```
import React from 'react'
import TodoInput from '../Todo/TodoInput'
import TodoList from '../Todo/TodoList'

export default function Todo() {
  return (
    <div>
      <TodoInput/>
      <TodoList/>
    </div>
  )
}
```

```
function App() {

  return (
    <div className="App container">
      <Router>
        <Routes>
          <Route path="/" element={<Todo />} />
        </Routes>
      </Router>
    </div>
  );
}
```

# TodoInput.jsx

TodoInput.jsx, kullanıcının yeni görevler eklemesi için bir arayüz sunar. Kullanıcı, metin girişi alanına yeni bir görev girer ve "Add new task" düğmesine tıkladığında bu görev sunucuya gönderilir. Yani, kullanıcı yeni görev eklemek istediğinde bu bileşen devreye girer. Özellikle to-do listesine yeni görevler eklemek için kullanılır.

```
import React, { useState } from 'react'
import './Todo.css';
import TodoService from '../../services/TodoService';
import { useNavigate } from 'react-router-dom';
export default function TodoInput() {
  const [todo, setTodo] = useState('');

  let todoService = new TodoService();
  const navigate = useNavigate()
  const addTodo = () => {
    if (todo.trim() !== '') {
      todoService.addTodo({ content: todo })
        .then((response) => {
          navigate(0)
        })
    }
  };

  return (
    <div className='container' >
      <h1>TodoInput</h1>
      <div className="card">
        <div className="card-body">
          <div className="input-group mb-3">
            <span className="input-group-text" id="basic-addon1"><i className="fa-solid fa-book"></i></span>
            <input
              type="text"
              className="form-control"
              placeholder="New todo"
              aria-label="New todo"
              aria-describedby="basic-addon1"
              value={todo}
              onChange={(e) => setTodo(e.target.value)}
            />
          </div>
          <div className="d-grid gap-2 ">
            <button className="btn btn-turk w-100" type="button" onClick={addTodo}>Add new task</button>
          </div>
        </div>
      </div>
    </div>
  )
}
```



```

import React, { useEffect, useState } from 'react'
import { useNavigate } from 'react-router-dom';
import './Todo.css';
import TodoService from '../../services/TodoService';

export default function TodoList() {
  let todoService = new TodoService()
  const navigate = useNavigate()
  const [isChecked, setIsChecked] = useState(false);
  const [update, setUpdate] = useState(false);
  const [todos, setTodos] = useState([])
  const [editableTodoId, setEditableTodoId] = useState(null);
  const [editedContent, setEditedContent] = useState('');
  const handleCheckboxChange = (todoId) => {
    setIsChecked(!isChecked);
    todoService.updateTodoIsDone(todoId)
      .then(() => {
        navigate(0);
      })
  };
  const handleUpdateContent = (todoId, content) => {
    todoService.updateTodoContent(todoId, content)
      .then(() => {
        navigate(0);
      })
  };
  const handleSetUpdateTrue = (todoId) => {
    setEditableTodoId(todoId);
  };
  const handleSetUpdateFalse = () => {
    setEditableTodoId(null);
  };
  const handleGetAllTodos = (e) => {
    todoService.getAllTodos().then(result => setTodos(result.data))
  };
  const handleGetAllTodosIsDoneTrue = (e) => {
    todoService.getAllTodosIsDoneTrue().then(result => setTodos(result.data))
  };
  const handleAllTodosIsDoneFalse = (e) => {
    todoService.getAllTodosIsDoneFalse().then(result => setTodos(result.data))
  };

```

```

const handleDeleteById = (e, todoId) => {
  var alert = window.confirm("Todo Silinsin mi ?");
  if (alert === true) {
    const res = todoService.deleteTodoById(todoId)
    navigate(0)
  } else {
    navigate(0)
  }
};
const handleDeleteAll = (e) => {
  var alert = window.confirm("Todolar Silinsin mi ?");
  if (alert === true) {
    const res = todoService.deleteTodosAll()
    navigate(0)
  }
};
const handleDeleteTodosIsDoneTrue = (e) => {
  var alert = window.confirm("Todolar Silinsin mi ?");
  if (alert === true) {
    const res = todoService.deleteTodosIsDoneTrue()
    navigate(0)
  }
};

useEffect(() => {
  todoService.getAllTodos().then(result => setTodos(result.data))
}, [])
return (
  <div className='container mt-3'>
    <h1>Todo List</h1>
    <div className="d-flex justify-content-between">
      <button className="btn btn-turk w-25" type="button" onClick={handleGetAllTodos}>All</button>
      <button className="btn btn-turk w-25" type="button" onClick={handleGetAllTodosIsDoneTrue}>Done</button>
      <button className="btn btn-turk w-25" type="button" onClick={handleAllTodosIsDoneFalse}>Todo</button>
    </div>
    {todos.length !== 0 ? (
      todos.map((todo) => (
        <div className="card mt-3" key={todo.id}>
          <div className="card-body d-flex justify-content-between align-items-center">
            {editableTodoId === todo.id ? (
              <div>
                <input
                  type="text"
                  id={todo.id}
                  value={editedContent}
                  className="m-0"

```

```

    onChange={() => setEditedContent(e.target.value)}
  />
  <a
    href="#"
    className="mx-3"
    onClick={() => handleUpdateContent(todo.id, editedContent)}
  >
    <i className="fa-regular fa-circle-check"></i>
  </a>
  <a href="#" className="mx-3" onClick={handleSetUpdateFalse}>
    <i className="fa-solid fa-circle-xmark"></i>
  </a>
</div>
) : (
  <input
    disabled
    type="text"
    style={
      todo.isDone
        ? { color: 'red', textDecoration: 'line-through' }
        : {}
    }
    value={todo.content}
    className="m-0"
  />
)}
<div>
  <input
    className="mx-3"
    type="checkbox"
    value=""
    id="flexCheckDefault"
    checked={todo.isDone}
    onChange={() => handleCheckboxChange(todo.id)}
  />
  <a
    href="#"
    className="mx-3"
    onClick={() => handleSetUpdateTrue(todo.id)}
  >
    <i className="fa-solid fa-pen"></i>
  </a>
  <a
    href="#"
    className="mx-3"
    onClick={(e) => handleDeleteById(e, todo.id)}
  >

```

```

    <i className="fa-solid fa-pen"></i>
  </a>
  <a
    href="#"
    className="mx-3"
    onClick={(e) => handleDeleteById(e, todo.id)}
  >
    <i className="fa-solid fa-trash"></i>
  </a>
</div>
</div>
))
): (
  <div className="card mt-3">
    <div className="card-body d-flex justify-content-between align-items-center">
      <p className="m-0">You don't have any todo here</p>
    </div>
  </div>
)}
<div className="d-flex justify-content-evenly">
  <button className="btn btn-delete w-50 m-3" type="button" onClick={handleDeleteTodosIsDoneTrue}>Delete done tasks</button>
  <button className="btn btn-delete w-50 m-3" type="button" onClick={handleDeleteAll}>Delete all tasks</button>
</div>
</div>
)
}

```




# FrontEnd

## Ekran Çıktısı

ve son olarak karşınızda  
projenin bitmiş hali  
bulunmaktadır.

### TodoInput



 New todo

Add new task



### Todo List

AllDoneTodo



Tech Career Full Stack Bootcamp

☒



Full Stack Developer Bitirme Projesi

☒

Redux Öğren

☐

React Native Öğren

☐

Delete done tasks

Delete all tasks