# Complete Repository Documentation

Repository: C:\Users\User\VisualStudio\Employee-Management-Sys\EmployeeManagementSystem

Generated: 2025-09-05 22:41:23

## Repository Overview

* \*\*Total Files:\*\* 15
* \*\*Languages:\*\* markdown, xml, java
* \*\*Project Type:\*\* maven\_java
* \*\*Frameworks:\*\*

## Repository Structure

EmployeeManagementSystem/

├── HELP.md

├── pom.xml

├── src/

├── main/

├── java/

├── com/

## Architecture Analysis

Detected Patterns: MVC/Layered Architecture, Repository Pattern, Data Transfer Object Pattern

Architectural Layers:

* \*\*Controller:\*\* 1 files
* \*\*Dto:\*\* 1 files
* \*\*Model:\*\* 1 files
* \*\*Repository:\*\* 1 files
* \*\*Service:\*\* 2 files
* \*\*Test:\*\* 1 files

## File Distribution

* \*\*Markdown:\*\* 1 files
* \*\*Xml:\*\* 1 files
* \*\*Java:\*\* 10 files

## Detailed File Documentation

### src\main\java\com\example\EmployeeManagementSystem\EmployeeManagementSystemApplication.java\n\*\*Language:\*\* Java\n\*\*Type:\*\* Main\n\n\*\*Classes:\*\*\n- `EmployeeManagementSystemApplication` (line 7)\n - `main()` (line 9)\n\n#### main\n# Function: main

## Description

The main function is the entry point for the Employee Management System application. It initializes the application and starts the execution.

## Parameters

* `args`: An array of strings representing command-line arguments passed to the application.

## Usage Example

public static void main(String[] args) {

EmployeeManagementSystemApplication application = new EmployeeManagementSystemApplication();

application.start();

}

## Important Notes

* This function should be called to start the Employee Management System application.
* Ensure that the necessary dependencies are properly configured before calling this function.\n\n---\n\n### src\main\java\com\example\EmployeeManagementSystem\controller\ControllerEmployee.java\n\*\*Language:\*\* Java\n\*\*Type:\*\* Controller\n\n\*\*Classes:\*\*\n- `ControllerEmployee` (line 19)\n - `createEmployee()` (line 25)\n - `findById()` (line 30)\n - `findAllEmployee()` (line 35)\n\n#### createEmployee\n# Function: createEmployee

## Description

This function is used to create a new employee in the Employee Management System.

## Parameters

* `dtaEmployee`: The data of the employee to be created.

## Usage Example

Employee newEmployee = new Employee("John Doe", "john.doe@example.com", "Manager");

createEmployee(newEmployee);

## Important Notes

* Make sure to provide all necessary information for the employee creation.
* Ensure that the data provided is valid and follows the required format.\n\n#### findById\n# findById Function Documentation

## Description

The findById function is used to retrieve an employee record from the Employee Management System by providing the employee's ID.

## Parameters

* `id`: The unique identifier of the employee whose record needs to be retrieved.

## Usage Example

Employee employee = findById(123);

System.out.println(employee);

## Important Notes

* This function will return an employee object based on the provided ID.
* If the ID does not exist in the system, a null value will be returned.
* Make sure to handle null returns appropriately in your code.\n\n---\n\n### src\main\java\com\example\EmployeeManagementSystem\dto\DtoEmployee.java\n\*\*Language:\*\* Java\n\*\*Type:\*\* Model\n\n\*\*Classes:\*\*\n- `DtoEmployee` (line 3)\n - `DtoEmployee()` (line 9)\n - `DtoEmployee()` (line 12)\n - `getId()` (line 19)\n\n#### getId\n# Function: getId

## Description

This function returns the ID of the employee.

## Parameters

None

## Usage Example

DtoEmployee employee = new DtoEmployee();

int id = employee.getId();

System.out.println("Employee ID: " + id);

## Important Notes

* This function is used to retrieve the ID of the employee from the DtoEmployee object.\n\n#### setId\n# Function: setId

## Description

This function sets the id of a DtoEmployee object to the specified value.

## Parameters

* id: The id to set for the DtoEmployee object.

## Usage Example

DtoEmployee employee = new DtoEmployee();

employee.setId(123);

## Notes

* This function is used to assign an id to a DtoEmployee object.
* Make sure to provide a valid id value as a parameter.\n\n---\n\n### src\main\java\com\example\EmployeeManagementSystem\exception\ExceptionEmployee.java\n\*\*Language:\*\* Java\n\*\*Type:\*\* Other\n\n\*\*Classes:\*\*\n- `ExceptionEmployee` (line 3)\n - `ExceptionEmployee()` (line 4)\n\n---\n\n### src\main\java\com\example\EmployeeManagementSystem\mapper\MapperEmployee.java\n\*\*Language:\*\* Java\n\*\*Type:\*\* Main\n\n\*\*Classes:\*\*\n- `MapperEmployee` (line 6)\n - `mapTOEmployee()` (line 7)\n - `mapToDtoEmployee()` (line 15)\n\n#### mapTOEmployee\n# Function: mapTOEmployee

## Description

This function maps a Data Transfer Object (DTO) representing an employee to an actual Employee object in the Employee Management System.

## Parameters

* `dtoEmployee`: The Data Transfer Object representing an employee that needs to be mapped to an Employee object.

## Usage Example

// Create a new DTO employee

DtoEmployee dtoEmployee = new DtoEmployee("John Doe", "john.doe@example.com", "Manager");

// Map the DTO employee to an Employee object

Employee employee = mapTOEmployee(dtoEmployee);

## Important Notes

* This function is located in the `MapperEmployee` class at line 7-12 in the specified file path.
* Ensure that the DTO employee has the necessary fields to be mapped to an Employee object.\n\n#### mapToDtoEmployee\n# mapToDtoEmployee

## Description

This function takes an employee object as input and maps it to a DTO (Data Transfer Object) representation of the employee.

## Parameters

* `employee`: The employee object to be mapped to a DTO.

## Usage Example

Employee employee = new Employee();

employee.setId(1);

employee.setName("John Doe");

employee.setDepartment("Engineering");

EmployeeDto employeeDto = mapToDtoEmployee(employee);

System.out.println(employeeDto.getName()); // Output: John Doe

## Important Notes

* This function is located in the MapperEmployee class at line 15-20 in the specified file path.\n\n---\n\n### src\main\java\com\example\EmployeeManagementSystem\model\Employee.java\n\*\*Language:\*\* Java\n\*\*Type:\*\* Model\n\n\*\*Classes:\*\*\n- `Employee` (line 9)\n - `Employee()` (line 19)\n - `Employee()` (line 22)\n - `getId()` (line 29)\n\n#### getId\n# Function: getId

## Description

This function returns the ID of the employee.

## Parameters

None

## Usage Example

Employee employee = new Employee();

int id = employee.getId();

System.out.println("Employee ID: " + id);

## Important Notes

* This function does not require any parameters as it retrieves the ID directly from the Employee object.\n\n#### setId\n# Function: setId

## Description

This function sets the ID of an employee in the Employee Management System.

## Parameters

* id: The unique identifier for the employee.

## Usage Example

Employee employee = new Employee();

employee.setId(12345);

## Notes

* This function is essential for properly identifying employees within the system.
* Ensure that the ID provided is unique and follows any specific guidelines set by the system.\n\n---\n\n### src\main\java\com\example\EmployeeManagementSystem\service\ServiceEmployee.java\n\*\*Language:\*\* Java\n\*\*Type:\*\* Service\n\n\*\*Classes:\*\*\n- `ServiceEmployee` (line 16)\n - `createEmployee()` (line 22)\n - `findById()` (line 29)\n - `findAllEmployee()` (line 36)\n\n#### createEmployee\n# Function: createEmployee

## Description

This function creates a new employee in the Employee Management System using the provided employee data transfer object (dtoEmployee).

## Parameters

* dtoEmployee: The data transfer object containing the information of the employee to be created.

## Usage Example

DtoEmployee employeeData = new DtoEmployee("John Doe", "john.doe@example.com", "Manager");

createEmployee(employeeData);

## Important Notes

* This function is responsible for adding a new employee to the system.
* Ensure that the dtoEmployee parameter contains valid information before calling this function.\n\n#### findById\n# findById Function Documentation

## Description

The findById function is used to retrieve an employee record from the Employee Management System database based on the provided employee ID.

## Parameters

* `id`: The unique identifier of the employee whose record needs to be retrieved.

## Usage Example

// Retrieve employee record with ID 123

Employee employee = serviceEmployee.findById(123);

System.out.println(employee);

## Important Notes

* Make sure to pass a valid employee ID as the parameter to retrieve the correct employee record.
* If the provided ID does not exist in the database, the function will return null.\n\n---\n\n

## Summary

* \*\*Total files in repository:\*\* 15
* \*\*Code files analyzed:\*\* 10
* \*\*Files documented:\*\* 7
* \*\*Languages detected:\*\* markdown, xml, java

Generated by Starter Doc Generator