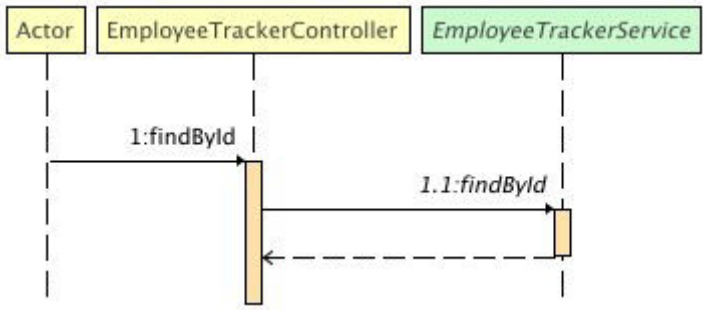
Employee Tracker Application Documentation

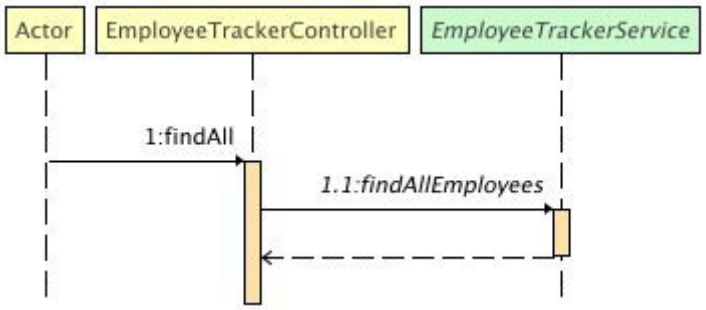
EmployeeTrackerController.class

Method: *findById(id)*



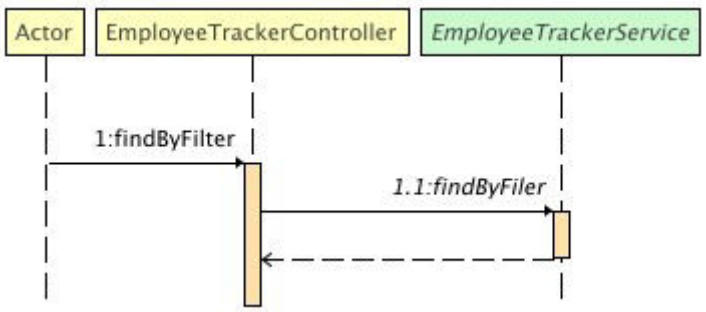
1. Client is calling ***findById()*** method and passing the id parameter to *api/employee-tracker/{id} path.* HTTP method is GET.
2. Controller is calling ***findById()*** method in LoanCalculatorService.class with provided id.

Method: *findAll()*



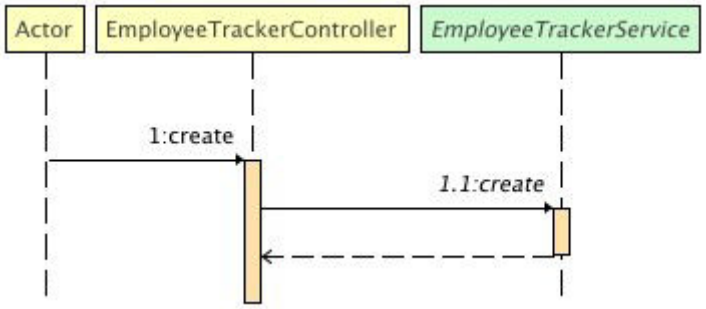
1. Client is calling ***findAll()*** method in controller at *api/employee-tracker/all path.* HTTP method is GET.
2. Controller is calling ***findAllEmployees()*** method in EmployeeTrackerService.class to retrieve all employees from database.

Method: *findByFilter(personalId, name, team, teamLead)*



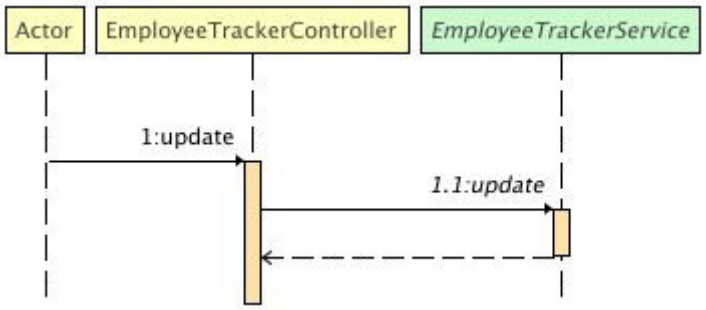
1. Client is calling ***findByFilter()*** method in controller at *api/employee-tracker/search path.* HTTP method is GET. Client can pass none, one or more than one parameter to this method for filtering results. If there is no passes parameters, method will retrieve all results from database.
2. Controller is calling ***findByFilter()*** method in EmployeeTrackerService.class to retrieve all employees from database with desired parameters.

Method: *create(employeeDTO)*



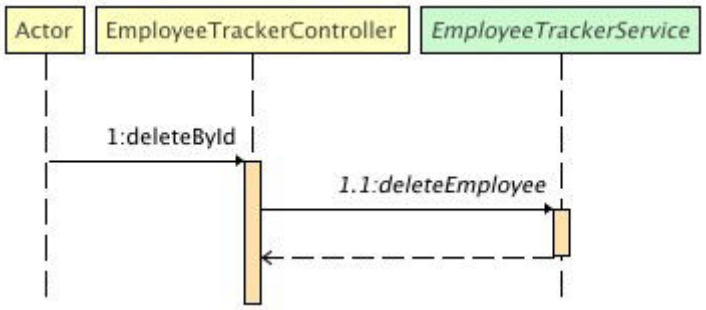
1. Client is calling ***create()*** method in controller at *api/employee-tracker/create path.* HTTP method is POST. As a parameter, client is passing the EmployeeDTO.class object.
2. Controller is calling ***create()*** method in EmployeeTrackerService.class create the new employee.

Method: *update(employeeDTO)*



1. Client is calling ***update()*** method in controller at *api/employee-tracker/update path.* HTTP method is PUT. As a parameter, client is passing the EmployeeDTO.class object.
2. Controller is calling ***update()*** method in EmployeeTrackerService.class update the existing employee.

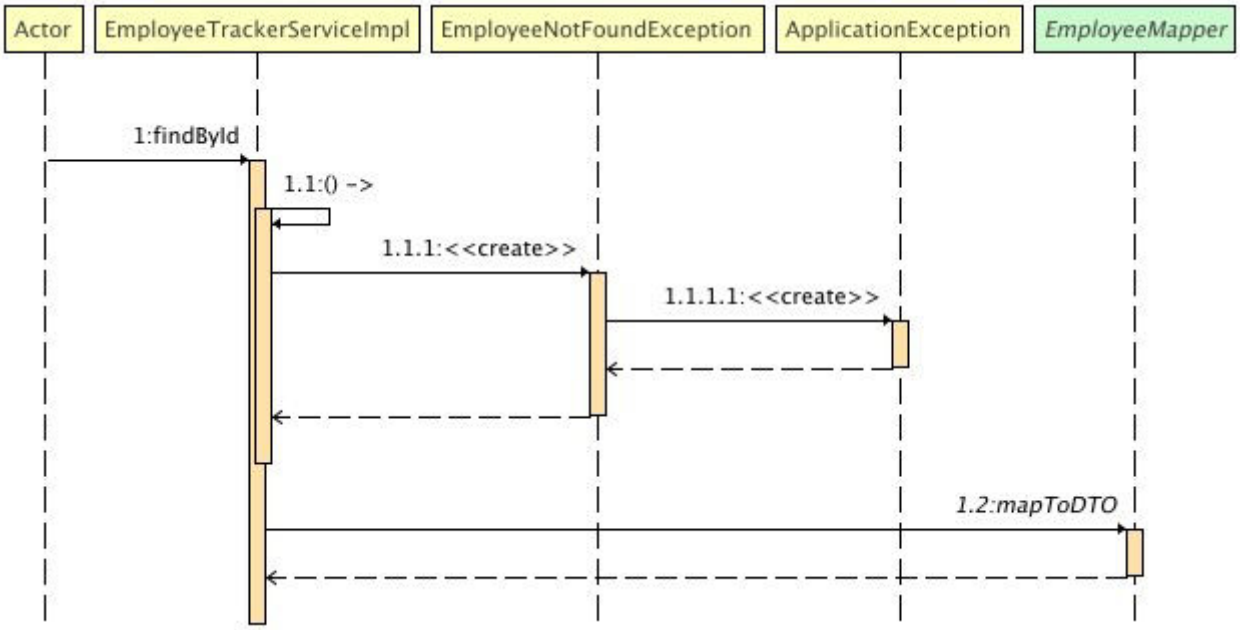
Method: *deleteById(id)*



1. Client is calling ***deleteById()*** method and passing the id parameter to *api/employee-tracker/{id} path.* HTTP method is DELETE.
2. Controller is calling ***deleteEmployee()*** method in LoanCalculatorService.class with provided id.

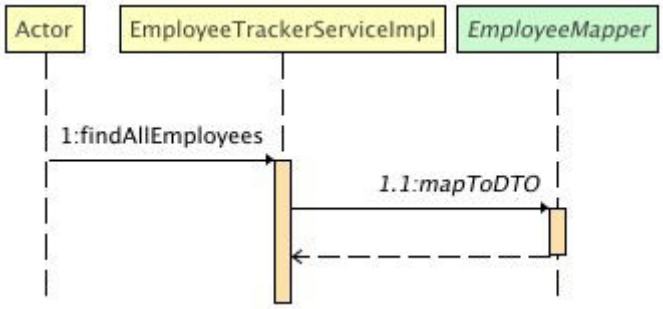
EmployeeTrackerService.class

Method: *findById(id)*



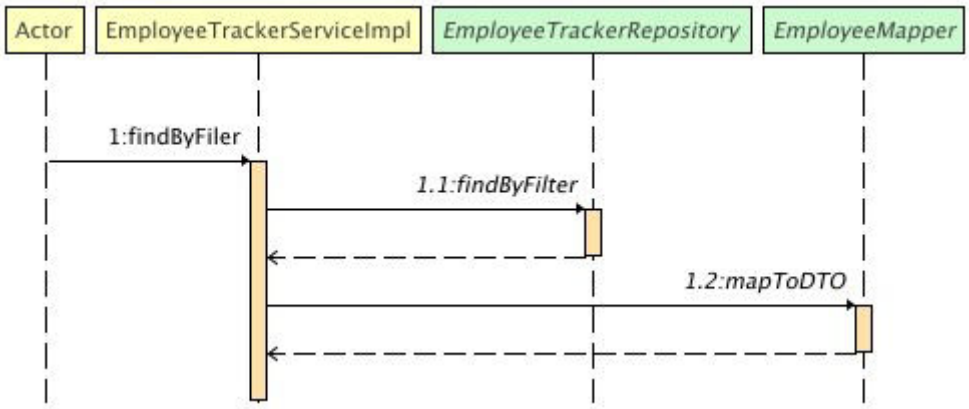
1. Controller is passing id from client to ***findById()*** method in EmployeeTrackerService.
2. In service class, we will use EmployeeTrackerRepository to call **findById()** method with provided id and expect to get Employee.class object from database as a result. Otherwise we will get EmployeeNotFoundException.
3. From retrieved request we will get Employee with provided id and with EmployeeMapper we will map them to Employee DTO object.

Method: *findAllEmployees()*



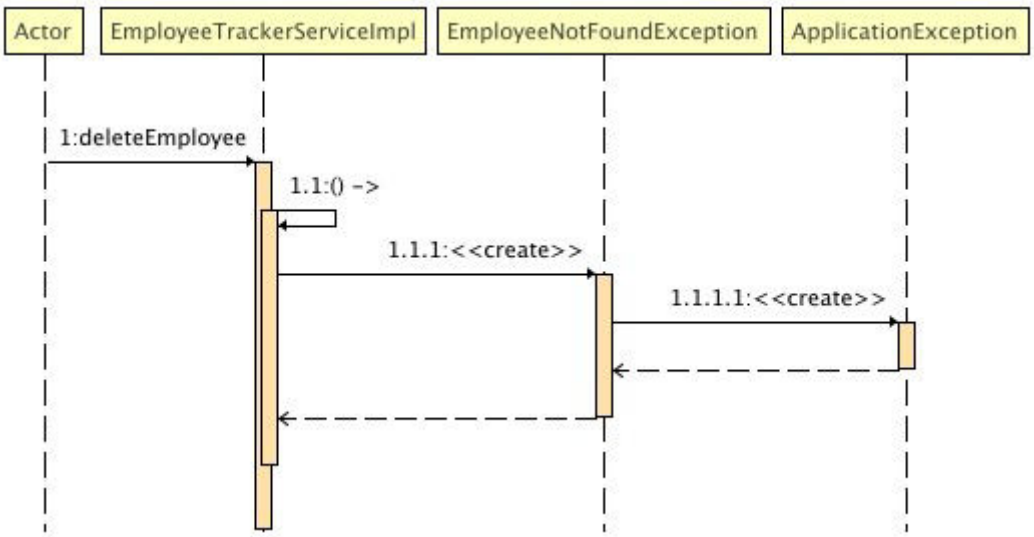
1. In this method at service class, we will use EmployeeTrackerRepository to call **findAll()** method and expect to get list of all Employee.class objects from database as a result.
2. As retrieved results we will get list of Employee objects which is going to be mapped with EmployeeMapper to list of EmployeeDTO objects.

Method: ***findAllByFilter(personalId, name, team, teamLead)***



1. In this method in service class, we will use EmployeeTrackerRepository to call custom method **findByFilter()** with or without parameters and expect to get list of all Employee.class objects from database as a result.
2. Custom method is based on query which will filter our results based on parameters.
3. As retrieved results we will get list of Employee objects which is going to be mapped with EmployeeMapper to list of EmployeeDTO objects.

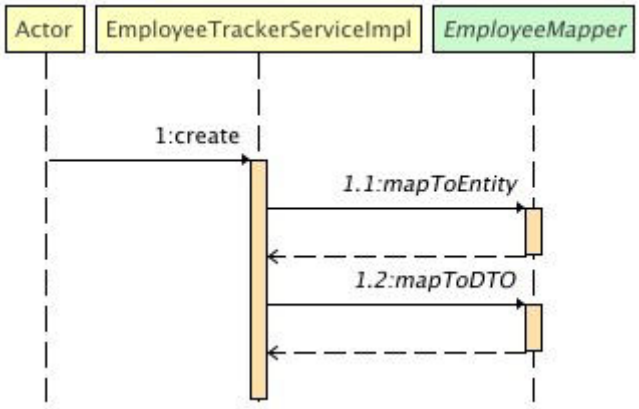
Method: ***deleteEmployee(id)***



1. Controller is passing id from client to ***deleteEmployee()*** method in EmployeeTrackerService.
2. In service class, we will use EmployeeTrackerRepository to call **findById()** method with provided id and expect to get Employee.class object from database as a result. Otherwise we will get EmployeeNotFoundException.

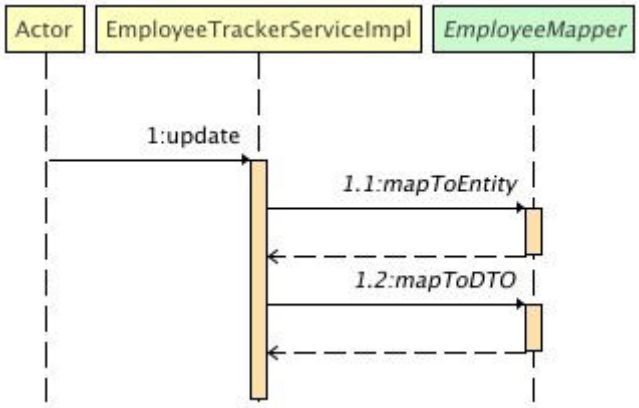
3. If there is a record in the database with provided id, repository will be called with ***deleteById()*** method and the record with provided id will be removed.

Method: *create(employeeDTO)*



1. As you can see in controller documentation, Controller is passing an employeeDTO object from client to ***create()*** method in EmployeeTrackerService.
2. In service class, with EmployeeMapper passed DTO object will be mapped to Employee.class entity object wi **mapToEntity()** method.
3. We will use EmployeeTrackerRepository to call **save()** method with provided entity object and save it to database.
4. Since we want to return saved object as a result, we will use ***mapToDto()*** method in EmployeeMapper.class to map saved employee and give it back as a result.

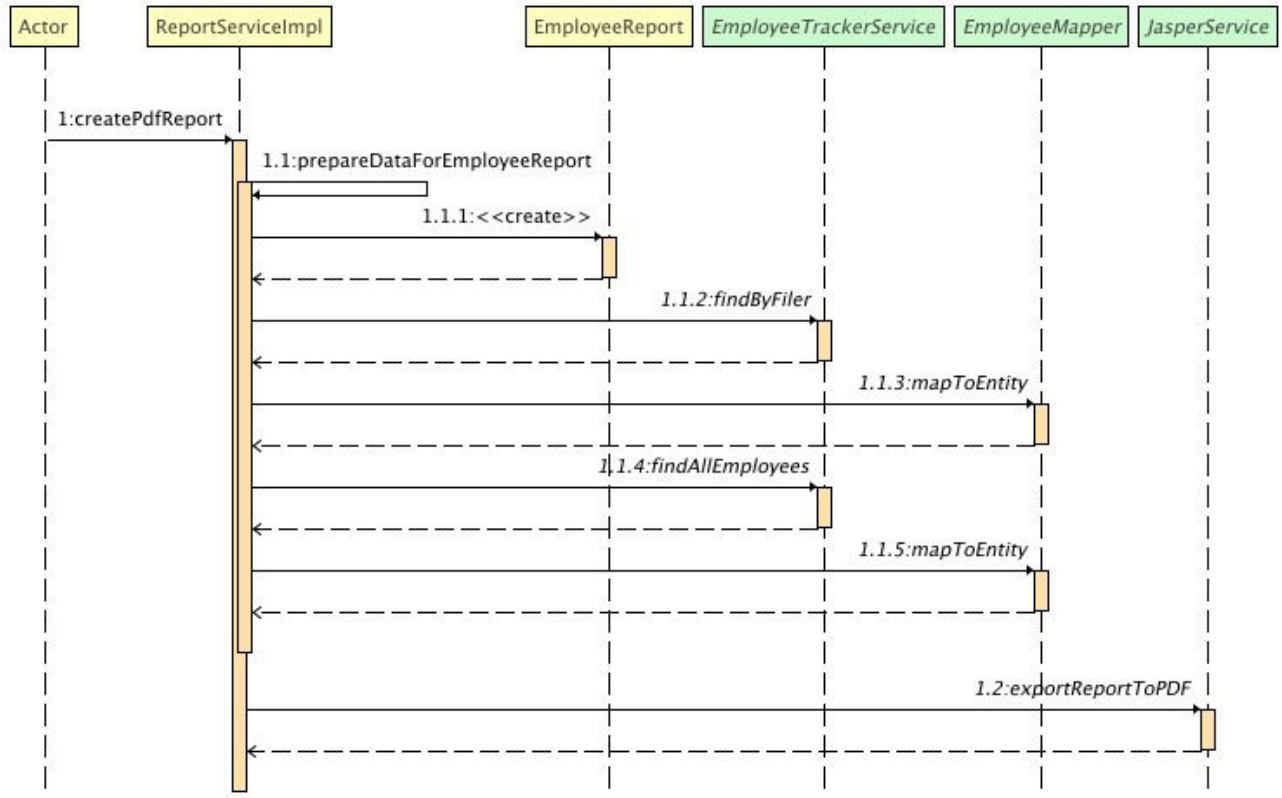
Method: update*(employeeDTO)*



1. Controller is passing an employeeDTO object from client to ***update()*** method in EmployeeTrackerService.
2. In service class, with EmployeeMapper passed DTO object will be mapped to Employee.class entity object wi **mapToEntity()** method.
3. For update purpose, we are setting passed id from DTO object to retrieved entity object.
4. We will use EmployeeTrackerRepository to call **save()** method with entity object and save it to database which will do the update.
5. Again, we want to return saved object as a result, we will use ***mapToDto()*** method in EmployeeMapper.class to map updated employee and give it back as a result.

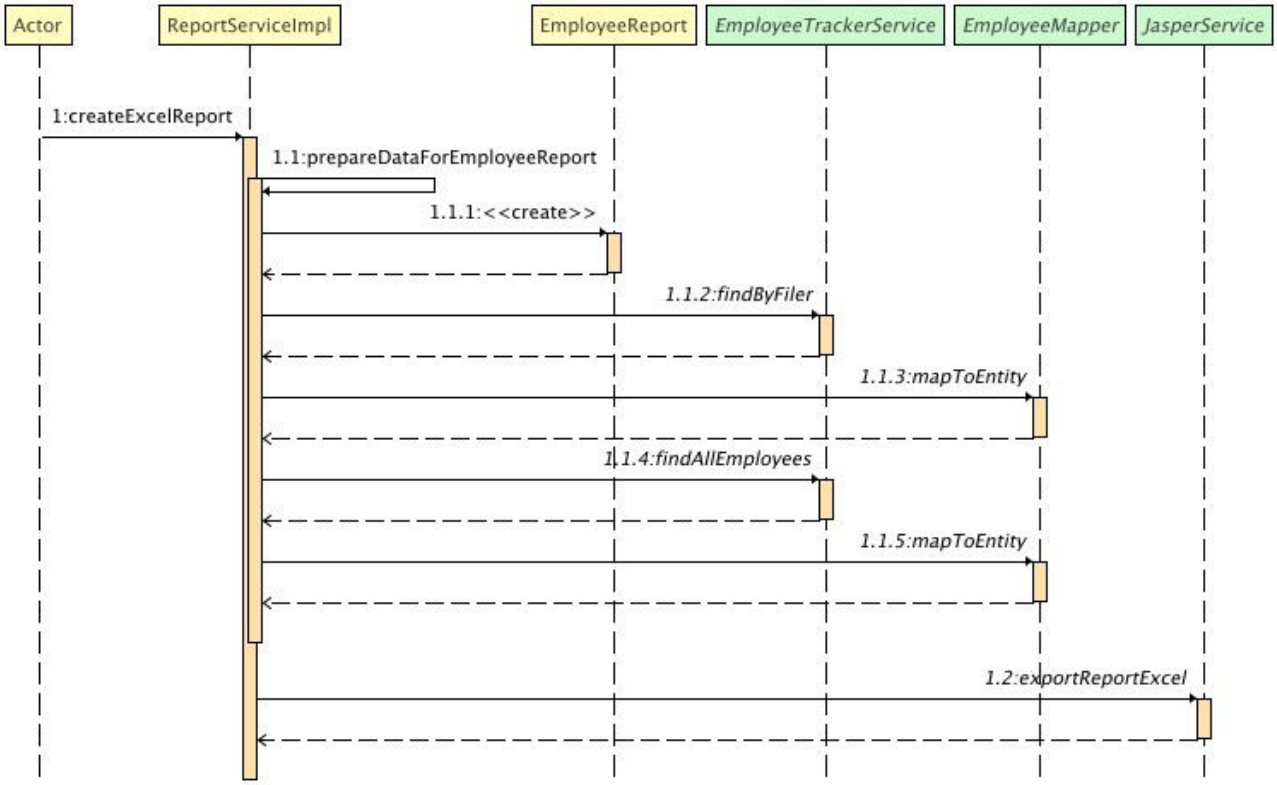
ReportController.class

Method: ***createPdfReport(team)***



1. Client is calling ***createPdfReport()*** method in ReportController.class for creating PDF report.
2. Here we have two cases. As a parameter we can pass the team String parameter or we can leave the method without any parameters. In dependence of that, we will filter the data for PDF report. The logic of report data filtering is implemented in private method ***prepareDataForEmployeeReport()*** in ReportServiceImpl.class. If we pass the **team** parameter, we will get the report of all employees from desired team and the ***findByFilter()*** method will be called with team parameter which is a good example of implementation task requirement for filter search method. If we leave it blank, method ***findAllEmployees()*** will be called we will get PDF report with all employees from our company.

Method: ***createExcelReport(team)***



1. Client is calling ***createExcelReport()*** method in ReportController.class for creating Excel report.
2. Here we have two cases. As a parameter we can pass the team String parameter or we can leave the method without any parameters. In dependence of that, we will filter the data for Excel report. The logic of report data filtering is implemented in private method ***prepareDataForEmployeeReport()*** in ReportServiceImpl.class. If we pass the **team** parameter, we will get the report of all employees from desired team and the ***findByFilter()*** method will be called with team parameter which is a good example of implementation task requirement for filter search method. If we leave it blank, method ***findAllEmployees()*** will be called we will get Excel report with all employees from our company.