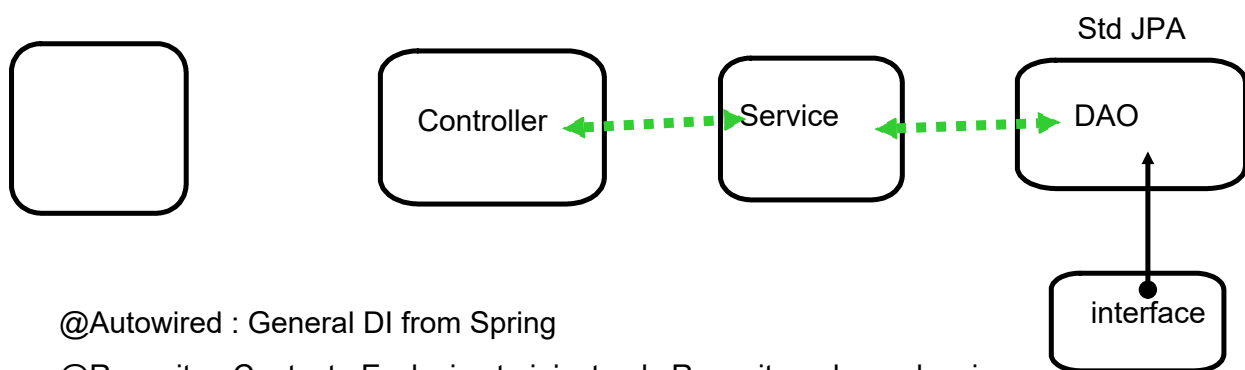Boot Starter JPA

  # USing entity manager (implementing native hibernate)

  # USing entity manager (standard JPA methods) : no vendor locking

  # Spring Boot Starter Data JPA (concrete implementation of CRUD functionalities)

Std JPA

Controller ⇄ Service ⇄ DAO

interface

@Autowired : General DI from Spring

@RepositoryContext : Exclusive to inject only Repository dependencies

TO add custom query requirements

    1. add / declare new method in interface having correct names

Sql

HQL

JPQL

    interface : <RepositoryName>Custom

StudentDaoCustom

Autowired

PersistentContext

PersistantContext : Exception arising by EM will be represented by
exception classes of JPA exception classes

   Student (DAO->Service->Controller) [CRUD Functionality]

   Employee (DAO->Service->Controller) [CRUD Functionality]

   Book (DAO->Service->Controller) [CRUD Functionality]

 Spring boot starter data rest

 addition of data-jpa project + rest project

   DaoRepository (interface inherits JPA Repository)

   DAO impl

   Service

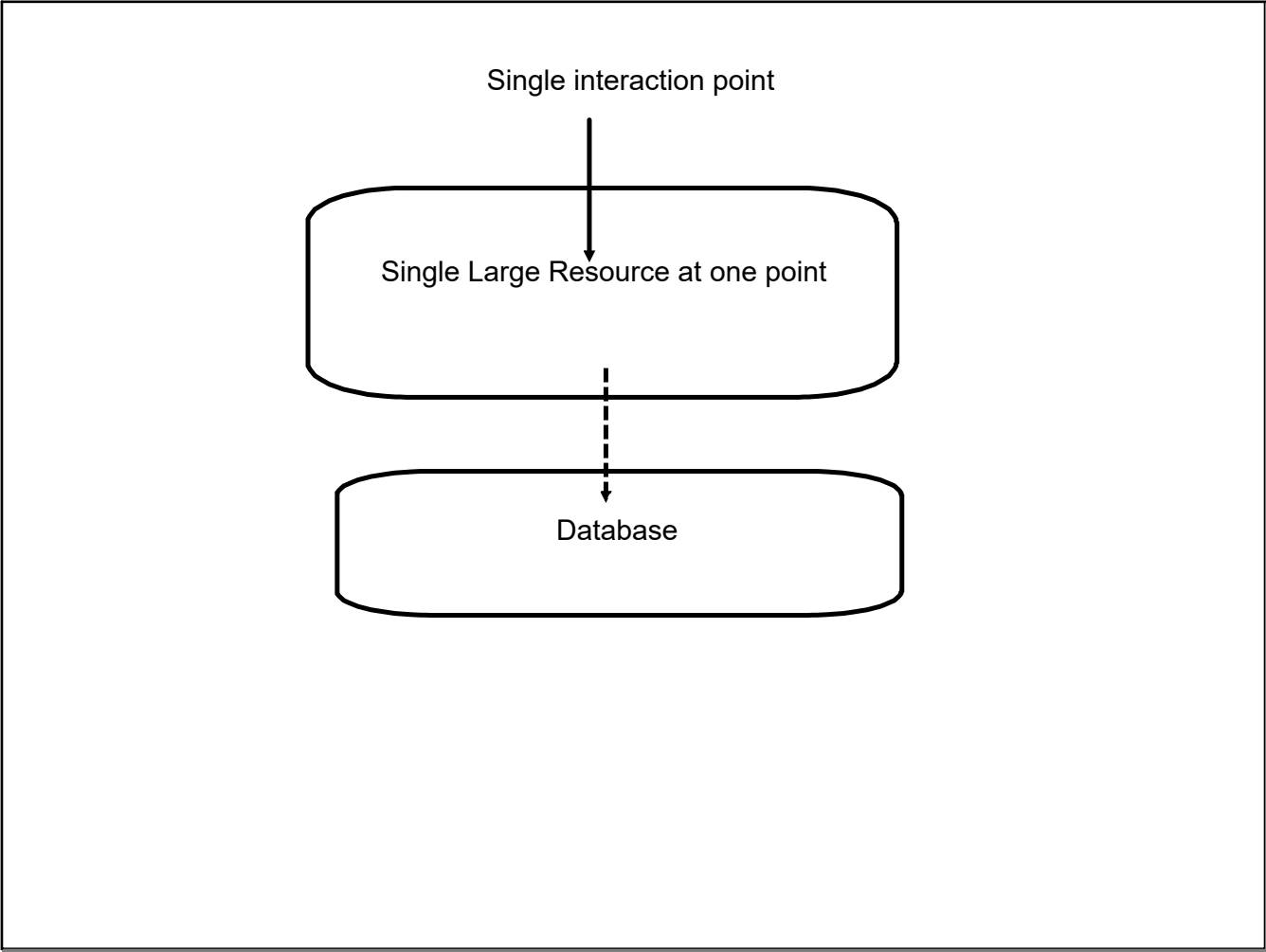   Controller

Rule

Entity : Student ~ plural

/students GET

/students/{studentId} GET

/student POST

/student PUT

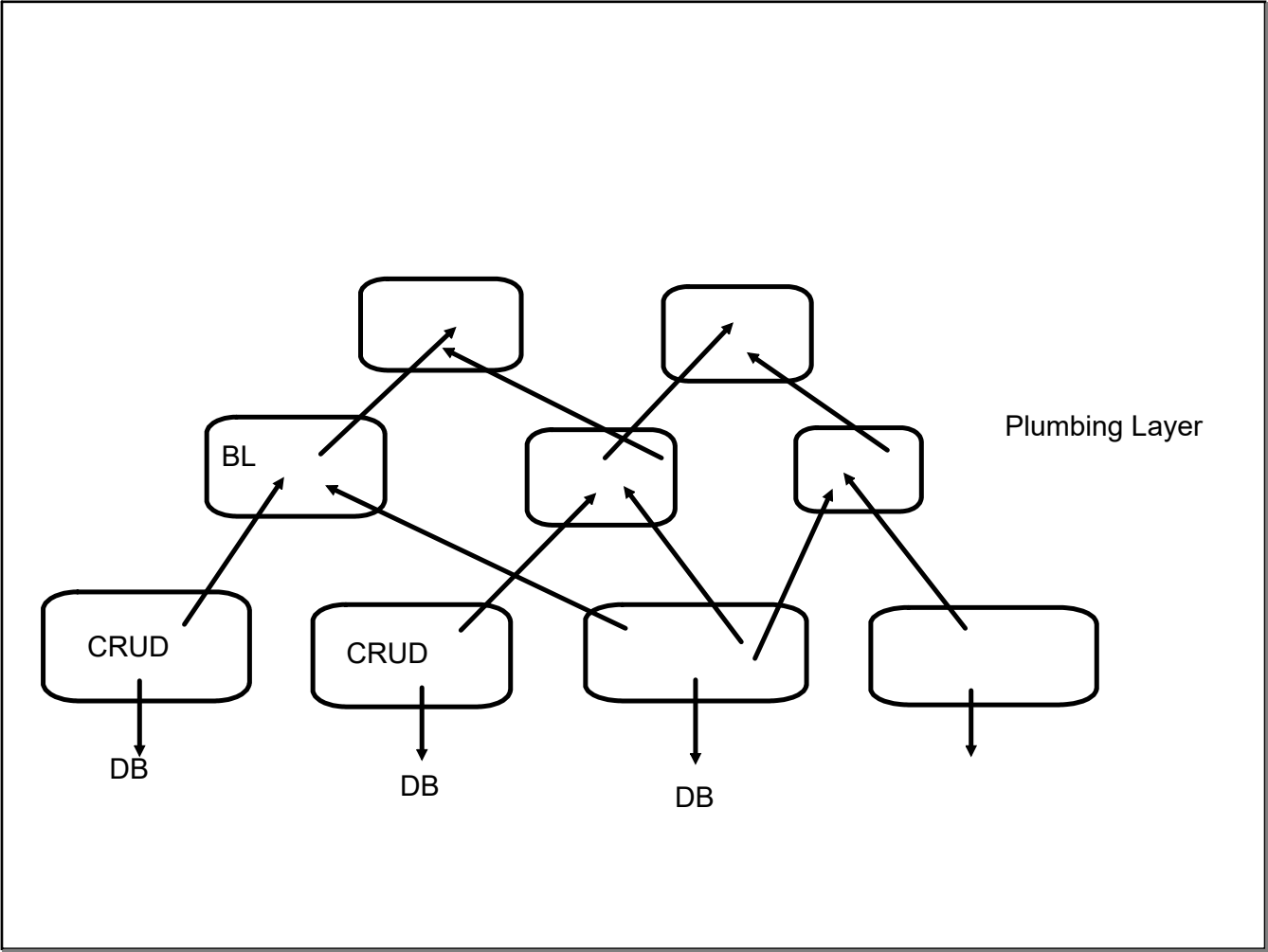/student/{studentId} DELETE

Spring Data REST : HATEOS Compliant

Single interaction point
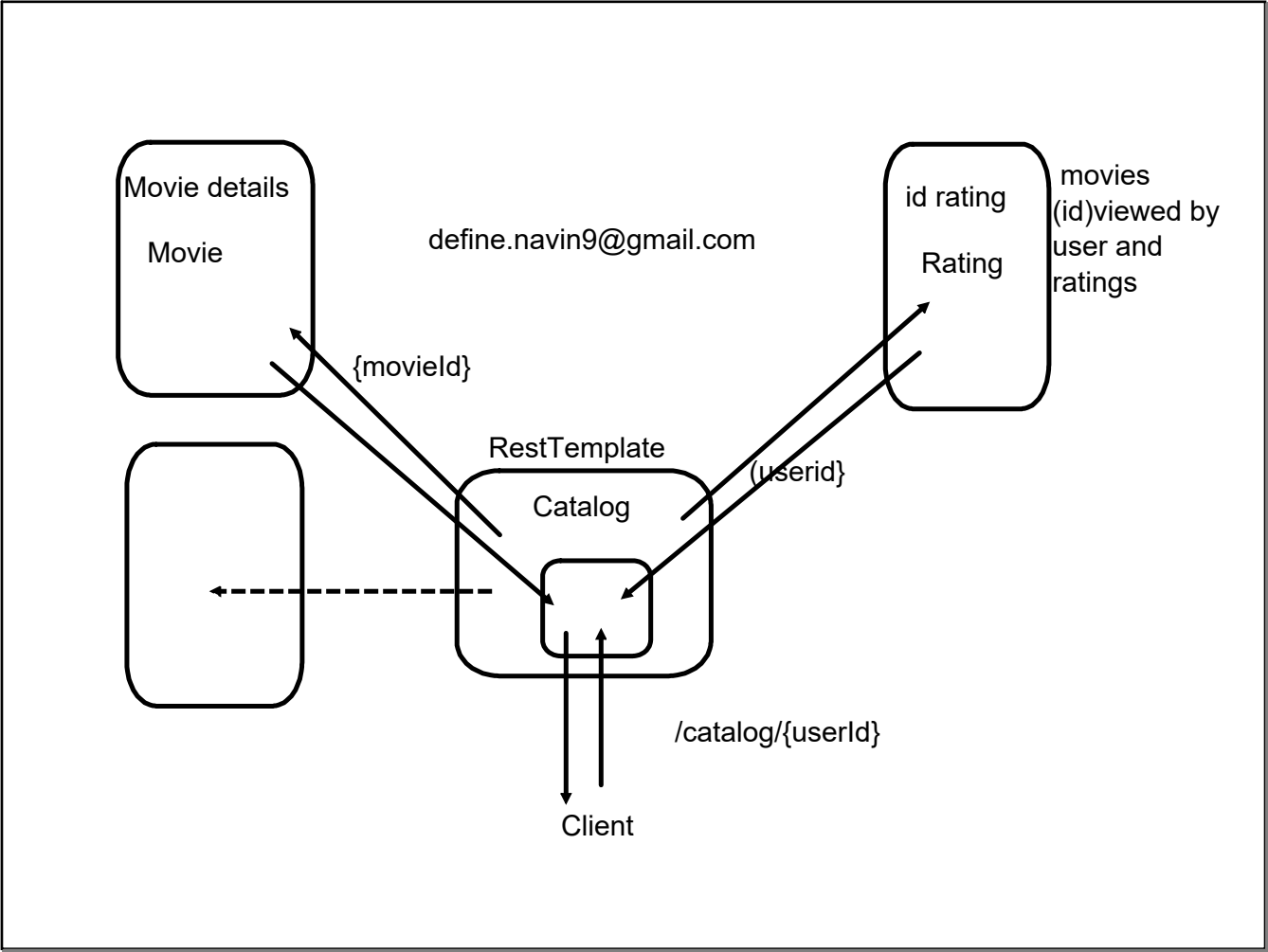
Single Large Resource at one point

Database

PHP

JAVA

Python

1. Not bounded to technology

2. Easy Debugging

Agile Methodology

3. Distributed running

4. Single module will not effect complete app

5. Spin up new instance when required

BL

Plumbing Layer

CRUD

CRUD

DB

DB

DB

Movie details

Movie

define.navin9@gmail.com

id rating

Rating

movies
(id)viewed by
user and
ratings

{movieId}

RestTemplate

Catalog

(userid}

/catalog/{userId}

Client

Collection

Don't return a collection

   1. Type specification on receiving end is complex

   2. Large data while streaming back can be a problem

   3. Future updation is not possible

Eureka
Discovery Server

9093

Eureka Client

9092

9091

Discovery Server

Catalog

movieMS

movie MS

AUTO

Eureka client

8761

Netflix