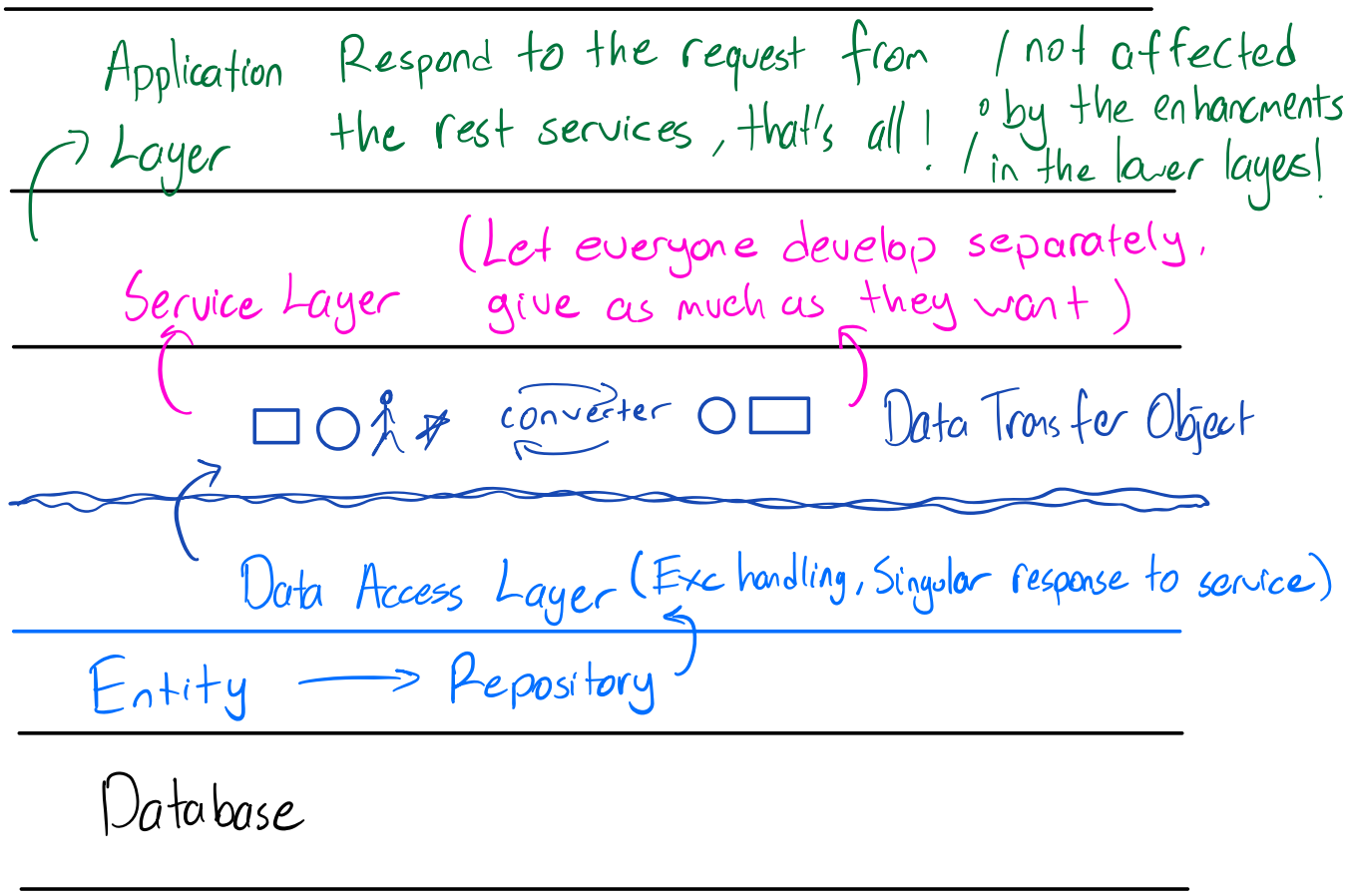
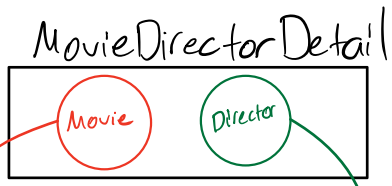


- what we basically do is,
  - → entities are transferred by repository in a list form.
  - → the methods of the entities from different repositories are implemented in Data Access Layer
  - → reached the service layer and data transfer object convert was done. (we are in RAM now!)
  - → controller (Application layer) responses to the requests sended to the service!



• entity

```
public class MovieDirectorDetail {
    private Movie m_movie;
    private Director m_director;

    public MovieDirectorDetail(Movie movie, Director director)
    {
        m_movie = movie;
        m_director = director;
    }
```

Movie Repository

Director Repository

• repository: example: I want to see the both the movies and the director. But the id number of the movie should also be hidden.

• example: implement methods

```
private static void fillMoviesDetails(ResultSet resultSet, ArrayList<MovieDirect
{
    do {
        var movieName : String = resultSet.getString( columnIndex: 1);
        var sceneTime : LocalDate = resultSet.getDate( columnIndex: 2).toLocalDate()
        var rating : long = resultSet.getLong( columnIndex: 3);
        var cost : BigDecimal = resultSet.getBigDecimal( columnIndex: 4);
        var directorName : String = resultSet.getString( columnIndex: 5);

        var movie = new Movie(movieName, sceneTime, rating, cost, k_imdb: 0);
        var director = new Director(directorName);
        details.add(new MovieDirectorDetail(movie, director));
    } while (resultSet.next());
```

Data Access Layer

• dat: filter the desired CRUD methods, deal with exceptions a kind of softening layer.

```
@Component
public class MovieServiceApplicationDAL {
    private final IMovieRepository m_movieRepository;
    private final IDirectorRepository m_directorRepository;

    public MovieServiceApplicationDAL(IMovieRepository movieRep
    {...}

    public Iterable<Movie> findAllMovies()
    {
        return doWorkForRepository(m_movieRepository::findAll
    }
```

Movie → convert → MovieDTO

Service Layer

example: let the movie-id number be shown only to the links I want

Application respond to the ⟹ rest service

http://192.168.1.101:9090/api/movie2/all

```
{
    "sceneTime": "2011-05-04",
    "imdb": 4.5,
    "id": 9,
    "rating": 23000,
    "cost": 822.00,
    "name": "the matrix"
},
```

Observe

```
private List<Movie> findAllMoviesCallbacky2()
{
    return StreamSupport.stream(k_movieServiceAppDAL.findAllMovies().spliterator(), parallel: false)
            .collect(Collectors.toList());
}

private List<MovieDTO> findMoviesByMonthYearCallback(int month, int year)
{
    return StreamSupport.stream(k_movieServiceAppDAL.findMoviesByMonthYear(month, year).spliterator(),
            .map(k_movieConverter::toMovieDTO)
            .collect(Collectors.toList());
}
```

http://192.168.1.101:9090/api/movie1/all

```
{
    "sceneTime": "04/05/2011",
    "rating": 23000,
    "cost": 822.00,
    "name": "the matrix"
},
```