

TUTORIAL 03 - MVC

ARSITEKTUR DAN PEMROGRAMAN APLIKASI PERUSAHAAN



Disusun oleh:

Riska Kurnia Dewi

1806269801

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS ILMU KOMPUTER

UNIVERSITAS INDONESIA

2020

Membuat Model

- Pada package model buatlah class MovieModel dengan spesifikasi sbb

```
MovieModel.java X
src > main > java > com > apap > tu03 > model > MovieModel.java > MovieModel
1 package com.apap.tu03.model;
2
3 public class MovieModel {
4     private String id;
5     private String title;
6     private String genre;
7     private Long budget;
8     private Integer duration;
9
10    public MovieModel(String id, String title, String genre, Long budget, Integer duration) {
11        this.id = id;
12        this.title = title;
13        this.genre = genre;
14        this.budget = budget;
15        this.duration = duration;
16    }
17
```

- Tambahkan method constructor, setter dan getter

```
public void setId(String id) {
    this.id = id;
}

public void setTitle(String title) {
    this.title = title;
}

public void setGenre(String genre) {
    this.genre = genre;
}

public void setBudget(Long budget) {
    this.budget = budget;
}

public void setDuration(Integer duration) {
    this.duration = duration;
}

public String getId() {
    return id;
}

public String getTitle() {
    return title;
}

public String getGenre() {
    return genre;
}

public Long getBudget() {
    return budget;
}

public Integer getDuration() {
    return duration;
}
```

Membuat Service

- Pada package service buatlah interface MovieService dengan spesifikasi sbb

MovieService.java

```
src > main > java > com > apap > tu03 > service >
1  package com.apap.tu03.service;
2
3  import java.util.List;
4  import com.apap.tu03.model.MovieModel;
5
6  public interface MovieService {
7      void addMovie(MovieModel movie);
8
9      List<MovieModel> getMovieList();
10
11     MovieModel getMovieDetail(String id);
12 }
13
```

- Pada package yang sama buatlah class InMemoryMovieService dengan spesifikasi sbb

InMemoryMovieService.java

```
src > main > java > com > apap > tu03 > service > InMemoryM
1  package com.apap.tu03.service;
2
3  import java.util.ArrayList;
4  import java.util.List;
5  import com.apap.tu03.model.MovieModel;
6  import org.springframework.stereotype.Service;
7
8  @Service
9  public class InMemoryMovieService implements MovieService {
10     private List<MovieModel> archiveMovie;
11
12     public InMemoryMovieService() {
13         archiveMovie = new ArrayList<>();
14     }
15
16     @Override
17     public void addMovie(MovieModel movie) {
18         archiveMovie.add(movie);
19     }
20
21     @Override
22     public List<MovieModel> getMovieList() {
23         return archiveMovie;
24     }
25
```

- Implementasikan method getMovieDetail

```
@Override
public MovieModel getMovieDetail(String id) {
    for (MovieModel movie : archiveMovie) {
        if (movie.getId().equals(id)) {
            return movie;
        }
    }
    return null;
}
```

Membuat Controller dan Method Add

- Pada package controller buatlah class MovieController dengan spesifikasi sbb

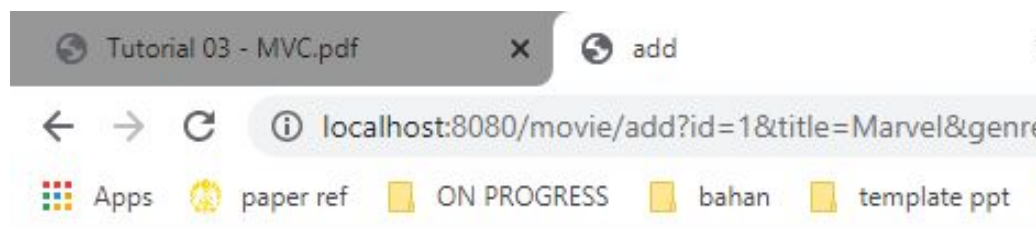
```
MovieController.java X
src > main > java > com > apap > tu03 > controller > MovieController.java
1  package com.apap.tu03.controller;
2
3  import com.apap.tu03.model.MovieModel;
4  import com.apap.tu03.service.MovieService;
5
6  import org.springframework.beans.factory.annotation.Autowired;
7  import org.springframework.stereotype.Controller;
8  import org.springframework.web.bind.annotation.RequestMapping;
9  import org.springframework.web.bind.annotation.RequestParam;
10
11  @Controller
12  public class MovieController {
13      @Autowired
14      private MovieService movieService;
15
16      @RequestMapping("/movie/add")
17      public String add(@RequestParam(value = "id", required = true) String id,
18                      @RequestParam(value = "title", required = true) String title,
19                      @RequestParam(value = "genre", required = true) String genre,
20                      @RequestParam(value = "budget", required = true) Long budget,
21                      @RequestParam(value = "duration", required = true) Integer duration) {
22          MovieModel movie = new MovieModel(id, title, genre, budget, duration);
23          movieService.addMovie(movie);
24          return "add";
25      }
26
27  }
```

- Buatlah view add.html dengan isi sbb

```
<> add.html X
src > main > resources > templates > <> add.html > html
1  <!DOCTYPE html>
2  <html xmlns:th="http://thymeleaf.org">
3
4  <head>
5  |   <title>add</title>
6  </head>
7
8  <body>
9  |   <h2>Data berhasil ditambahkan!</h2>
10 </body>
11 |
12 </html>
```

- Run project lalu akses

<http://localhost:8080/movie/add?id=1&title=Marvel&genre=Action&budget=50000000&duration=50>



Data berhasil ditambahkan!

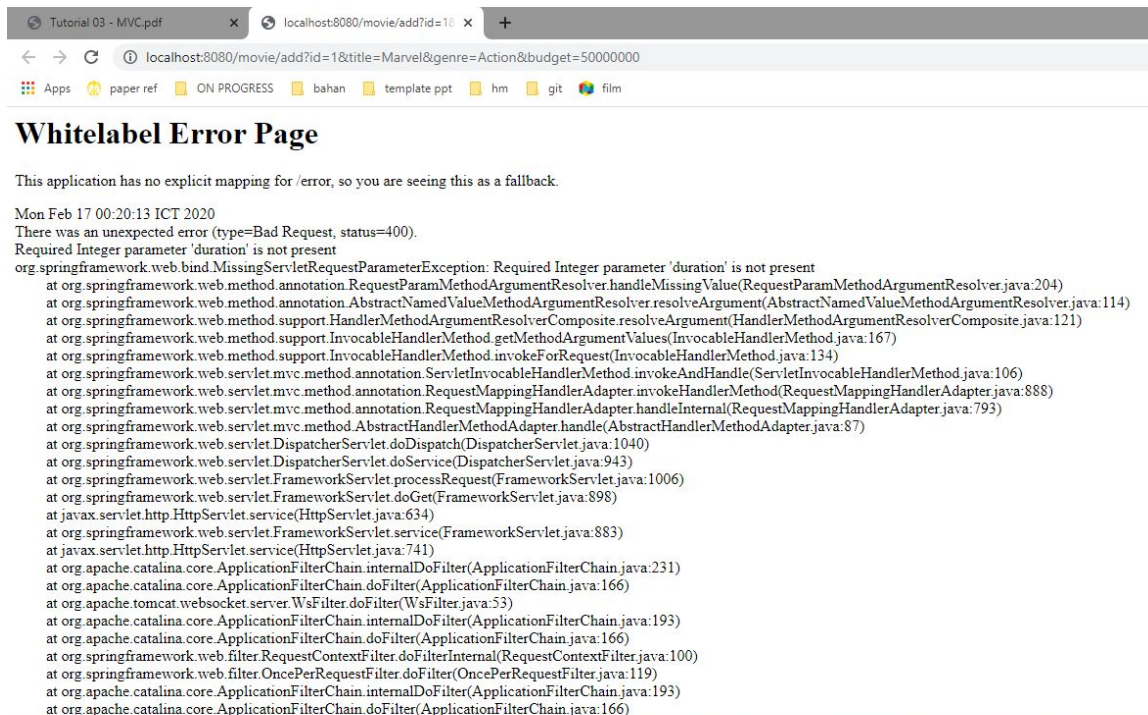
Q1: Apakah hasilnya? Jelaskan apa yang terjadi (lengkapi dengan screen capture)

A1: Ketika mengakses link tersebut, browser akan menampilkan kalimat "Data berhasil ditambahkan!". Hal ini terjadi karena sebelumnya pada class MovieController telah dilakukan mapping url yang akan me-return data ke view add.html, yaitu /movie/add. Url tersebut membutuhkan parameter id, title, genre budget dan duration dimana data tersebut diambil dari method constructor yang ada pada class MovieModel. Pada

MovieModel, terdapat method setter dan getter sehingga output yang dihasilkan sesuai dengan inputan. Kemudian MovieModel tersebut ditambahkan ke dalam MovieService.

- Akses

<http://localhost:8080/movie/add?id=1&title=Marvel&genre=Action&budget=50000000>



Q2: Apakah hasilnya? Jelaskan apa yang terjadi (lengkapi dengan screen capture).

A2: Terjadi error karena jumlah parameter yang diinputkan tidak sesuai dengan jumlah parameter yang dibutuhkan. Pada MovieController membutuhkan parameter id, title, genre, budget dan duration. Sedangkan saat mengakses link, tidak terdapat duration sehingga akan terjadi error.

Membuat Method View by ID

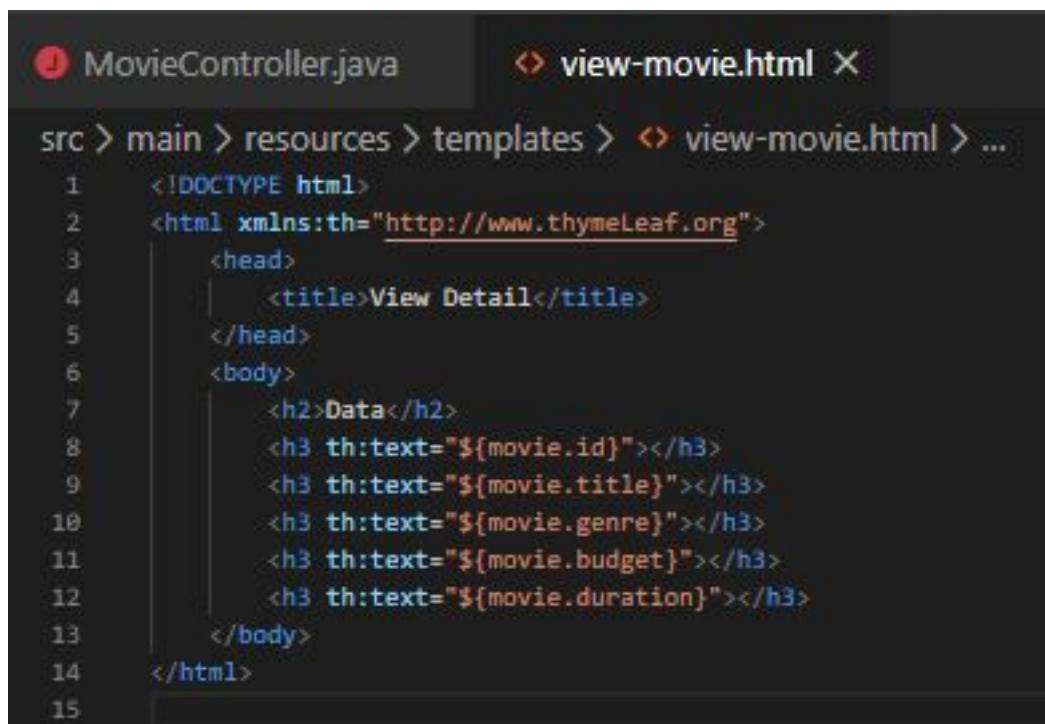
- Pada class MovieController tambahkan method berikut

```

@RequestMapping("/movie/view")
public String view(@RequestParam("id") String id, Model model){
    MovieModel archive = movieService.getMovieDetail(id);
    model.addAttribute("movie", archive);
    return "view-movie";
}

```

- Buatlah view-movie.html dengan isi sbb

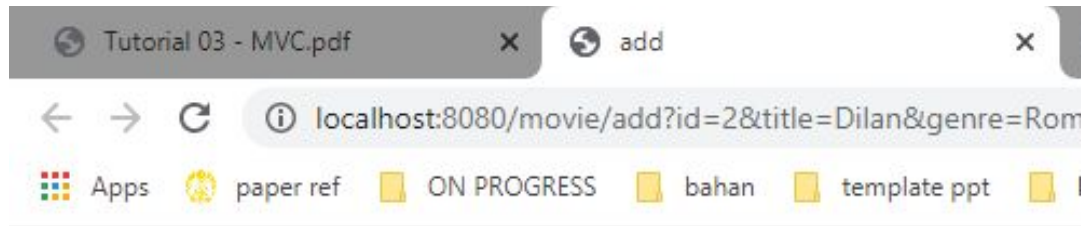


```

src > main > resources > templates > view-movie.html > ...
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>View Detail</title>
5      </head>
6      <body>
7          <h2>Data</h2>
8          <h3 th:text="${movie.id}"></h3>
9          <h3 th:text="${movie.title}"></h3>
10         <h3 th:text="${movie.genre}"></h3>
11         <h3 th:text="${movie.budget}"></h3>
12         <h3 th:text="${movie.duration}"></h3>
13     </body>
14 </html>
15

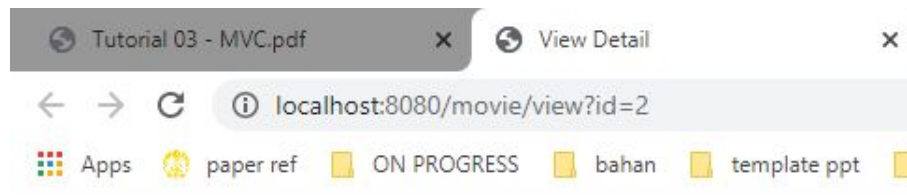
```

- Run project lalu akses
<http://localhost:8080/movie/add?id=2&title=Dilan&genre=Romance&budget=10000000&duration=110>



Data berhasil ditambahkan!

- Akses <http://localhost:8080/movie/view?id=2>



Data

2

Dilan

Romance

10000000

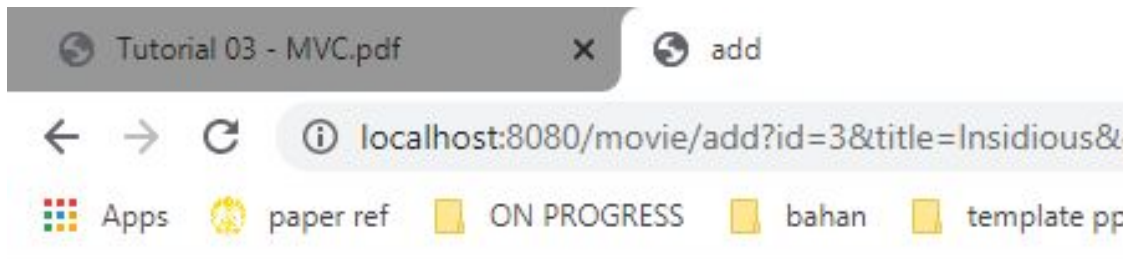
110

Q3: Apakah data movie tersebut muncul? Jelaskan apa yang terjadi (lengkapi dengan screen capture).

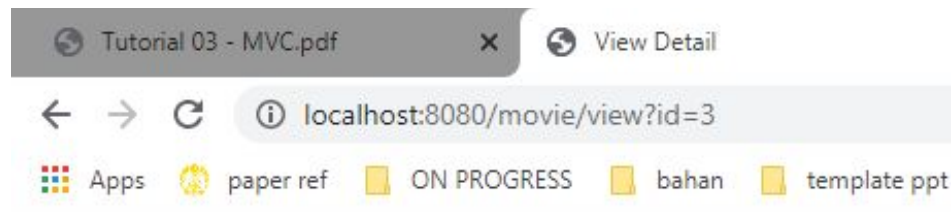
A3: Data yang telah ditambahkan sebelumnya muncul di browser. Hal ini disebabkan karena data tersebut ditambahkan ke archiveMovie. Kemudian pada saat mengakses <http://localhost:8080/movie/view?id=2> maka akan mengakses method view pada MovieController dimana parameter yang diberikan adalah 2. Lalu melakukan pencarian id bernilai 2 pada archiveMovie. Apabila data yang dicari sesuai dengan data yang telah ditambahkan sebelumnya, maka data tersebut akan direturn ke view view-movie.

- Tambahkan data movie lainnya dengan id yang berbeda

<http://localhost:8080/movie/add?id=3&title=Insidious&genre=Horor&budget=25000000&duration=130>



Data berhasil ditambahkan!



Data

3

Insidious

Horor

25000000

130

Membuat Method View All

- Pada class MovieController tambahkan method berikut

```
@RequestMapping("/movie/viewall")
public String viewAll(Model model){
    List<MovieModel> archive = movieService.getMovieList();
    model.addAttribute("movies", archive);
    return "viewall-movie";
}
```

- Buatlah viewall-movie.html dengan isi sbb

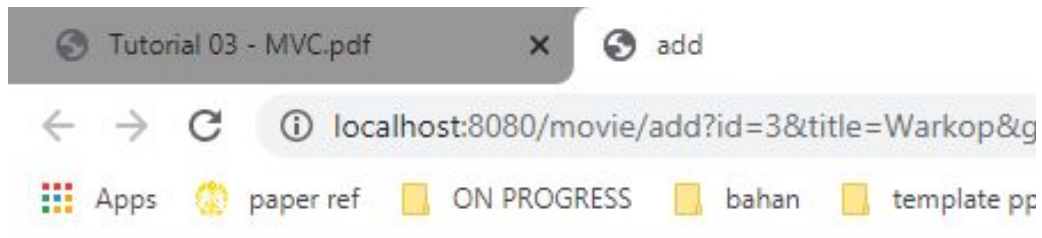


The screenshot shows an IDE with two tabs: 'MovieController.java' and 'viewall-movie.html'. The 'viewall-movie.html' tab is active, displaying the following HTML code:

```
src > main > resources > templates > viewall-movie.html > ...
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3     <head>
4         <title>View Detail</title>
5     </head>
6     <body>
7         <h2>Data</h2>
8         <div th:each="movie : ${movies}">
9             <h3 th:text="${movie.id}"></h3>
10            <h3 th:text="${movie.title}"></h3>
11            <h3 th:text="${movie.genre}"></h3>
12            <h3 th:text="${movie.budget}"></h3>
13            <h3 th:text="${movie.duration}"></h3>
14        </div>
15    </body>
16 </html>
17
```

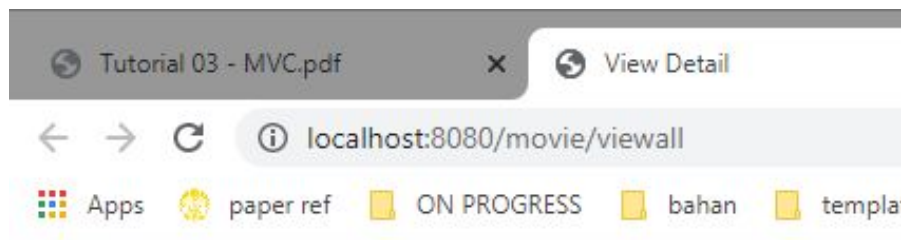
- Run project lalu akses

<http://localhost:8080/movie/add?id=3&title=Warkop&genre=Comedy&budget=30000000&duration=70>



Data berhasil ditambahkan!

- Akses <http://localhost:8080/movie/viewall>



Data

3

Warkop

Comedy

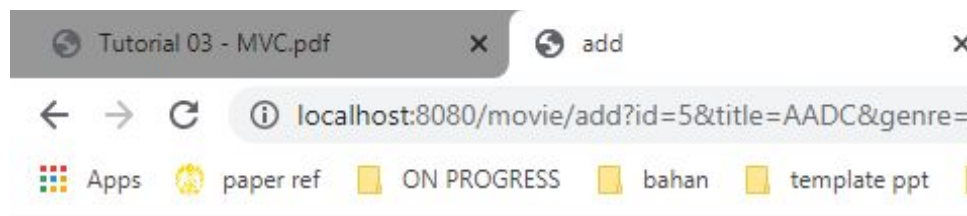
30000000

70

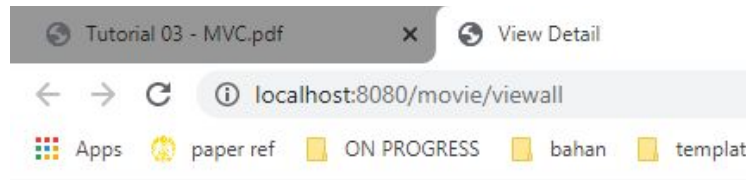
Q4: Apakah data movie tersebut muncul? Jelaskan apa yang terjadi (lengkapi dengan screen capture)

A4: Data akan dimunculkan di browser. Pada saat mengakses `http://localhost:8080/movie/viewall`, `viewAll` pada `MovieController` akan dijalankan. Method ini akan mengakses `archiveMovie` yang bertujuan untuk mengambil seluruh data yang telah ditambahkan sebelumnya. Lalu data yang telah diambil akan direturn ke view `viewall-movie`. Pada `viewall-movie` terdapat iterasi yang digunakan untuk menampilkan data array yang telah direturn dari controller..

- Tambahkan data movie lainnya dengan id yang berbeda (`http://localhost:8080/movie/add?id=5&title=AADC&genre=Romance&budget=450000000&duration=120`) lalu akses `http://localhost:8080/movie/viewall`



Data berhasil ditambahkan!



Data

3

Warkop

Comedy

30000000

70

5

AADC

Romance

450000000

120

Q5: Apakah semua data movie muncul? Jelaskan apa yang terjadi (lengkapi dengan screen capture).

Seluruh data movie yang telah ditambah akan ditampilkan ke browser karena pada viewall-movie telah dilakukan iterasi dengan tujuan untuk menampilkan seluruh data dari array.

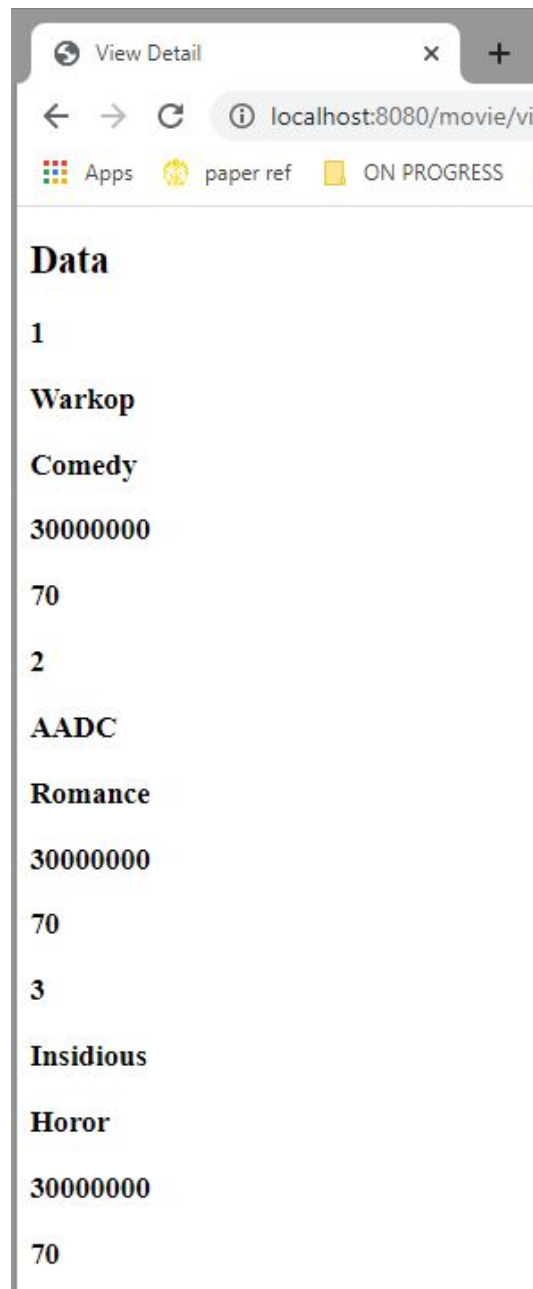
Latihan

1. Pada MovieController tambahkan sebuah method menampilkan movie dengan menggunakan PathVariable. Misalnya, Anda memasukkan data sebuah movie yang memiliki id 1, untuk melihat data yang baru dimasukkan tersebut maka dilakukan dengan mengakses halaman: localhost:8080/movie/view/1 Jika nomor id tidak diberikan atau tidak ditemukan, kembalikan halaman error yang berisi informasi bahwa nomor id kosong atau tidak ditemukan.

```
@RequestMapping(value= {"/movie/view","movie/view/{id}"})
public String view(@PathVariable Optional<String> id, Model model){
    if(id.isPresent()){
        MovieModel archive = movieService.getMovieDetail(id.get());
        if(archive == null){
            model.addAttribute("errorMessage", "ID not found");
            return "errPage";
        }else{
            model.addAttribute("movie", archive);
            return "view-movie";
        }
    }else{
        model.addAttribute("errorMessage", "ID is empty");
        return "errPage";
    }
}
```

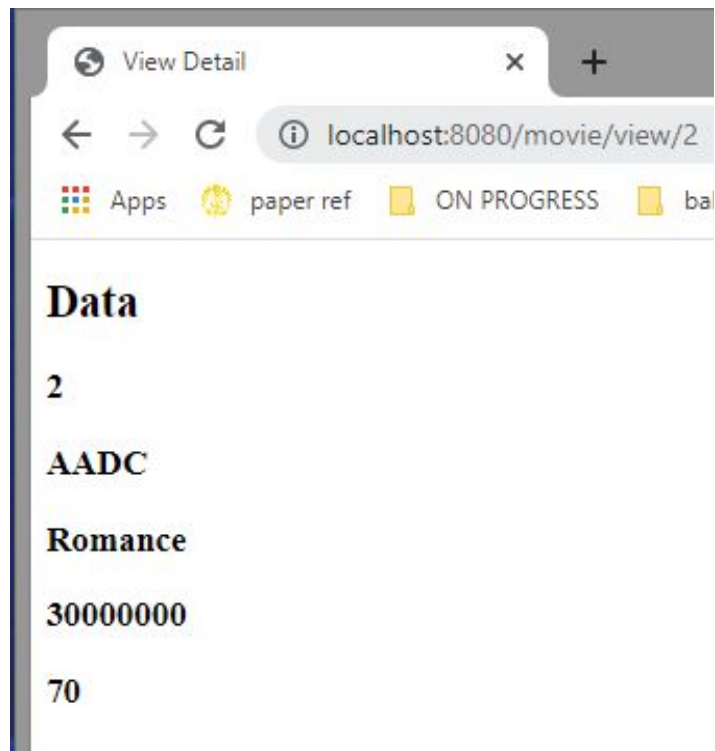
Pada MovieController ditambahkan method view dengan menggunakan Path Variable untuk menghandle request pada URL /movie/view/{id}. Jika id bernilai null atau parameter get id tidak ditemukan maka akan dialihkan ke view errPage.html dengan mengirimkan atribut errorMessage sesuai dengan error yang ada, yaitu ID tidak ditemukan atau ID kosong.

Berikut ini merupakan seluruh data yang telah ditambahkan.

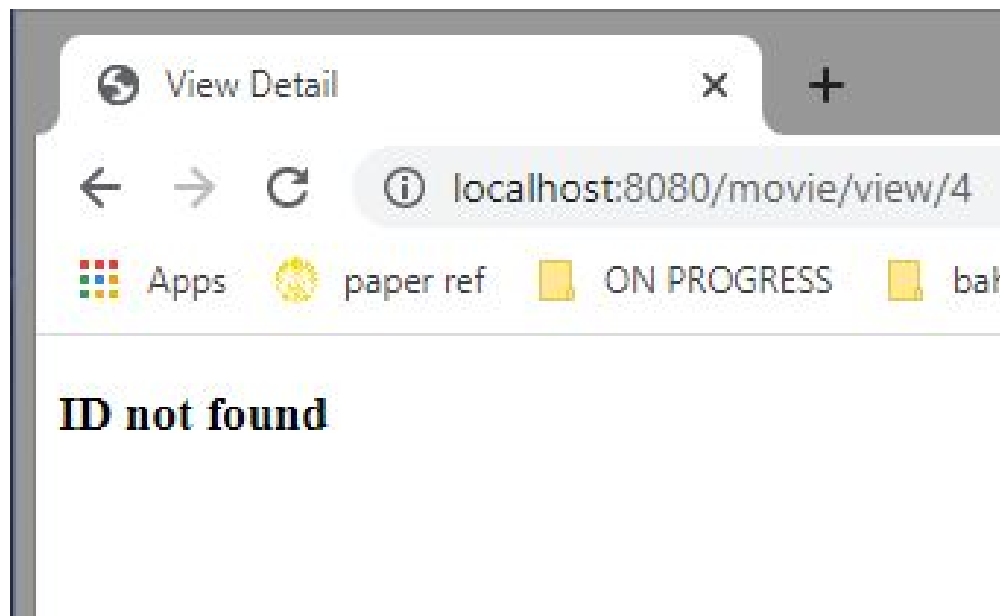


Data			
1	Warkop	Comedy	30000000
70			
2	AADC	Romance	30000000
70			
3	Insidious	Horror	30000000
70			

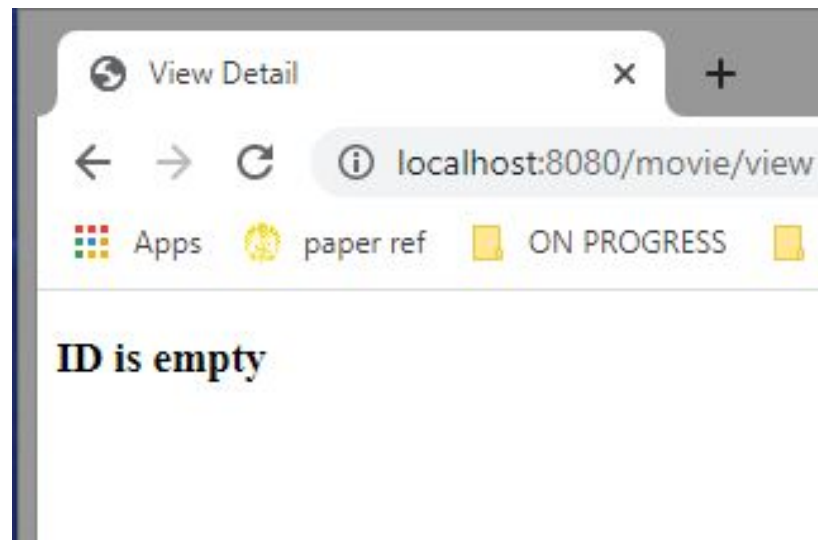
Apabila data sesuai dengan request yang diterima maka akan ditampilkan ke view view-movie seperti di bawah ini.



Namun, apabila data tidak sesuai dengan data yang telah ditambahkan sebelumnya maka akan menampilkan pesan error "ID not found".



Kemudian apabila tidak terdapat parameter id pada url yang diakses maka akan memunculkan pesan error "ID is empty".



2. Tambahkan fitur untuk melakukan update duration dari movie berdasarkan id. Misalnya, setelah melakukan add movie pada soal nomor 1, cobalah untuk mengubah duration objek movie tersebut menjadi 100 dengan mengakses halaman: localhost:8080/movie/update/1/duration/100. Tampilkan sebuah halaman yang memberikan informasi bahwa data tersebut telah berhasil diubah. Jika nomor id tidak diberikan atau tidak ditemukan, kembalikan halaman error yang berisi informasi bahwa nomor id kosong atau tidak ditemukan dan proses update dibatalkan.

MovieService.java X InMemoryMovieService.java MovieCon

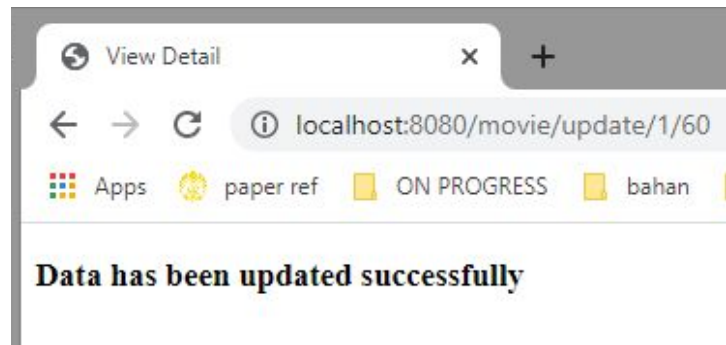
src > main > java > com > apap > tu03 > service > MovieService.java > ...

```
1 package com.apap.tu03.service;
2
3 import java.util.List;
4 import com.apap.tu03.model.MovieModel;
5
6 public interface MovieService {
7
8     List<MovieModel> getMovieList();
9
10    MovieModel getMovieDetail(String id);
11
12    void addMovie(MovieModel movie);
13
14    void deleteMovie(MovieModel movie);
15
16    void updateMovie(MovieModel movie, int duration);
17 }
```

```
@Override
public void updateMovie(MovieModel movie, int duration){
    for(int i=0; i<archiveMovie.size(); i++){
        if(archiveMovie.get(i).equals(movie)){
            archiveMovie.get(i).setDuration(duration);
        }
    }
}
```

```
@RequestMapping(value={"/movie/update", "movie/update/{id}/{duration}"})
public String update(@PathVariable Optional<String> id, @PathVariable Optional<Integer> duration, Model model){
    if(id.isPresent() && duration.isPresent()){
        MovieModel archive = movieService.getMovieDetail(id.get());
        if(archive == null){
            model.addAttribute("errMessage", "Data not found");
            return "errPage";
        }else{
            model.addAttribute("errMessage", "Data has been updated successfully");
            movieService.updateMovie(archive, duration.get());
            return "errPage";
        }
    }else{
        model.addAttribute("errMessage", "Data is empty");
        return "errPage";
    }
}
```

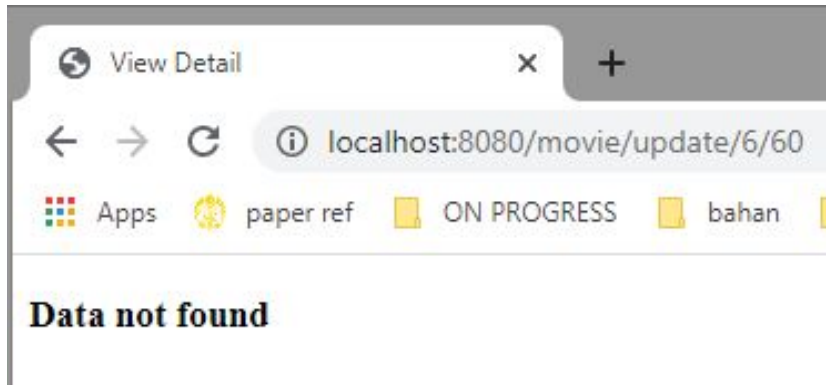

Apabila id yang menjadi parameter sesuai dengan data yang ditambahkan, maka data duration yang sesuai dengan id tersebut akan diubah. Kemudian akan muncul pesan "Data has been updated successfully".



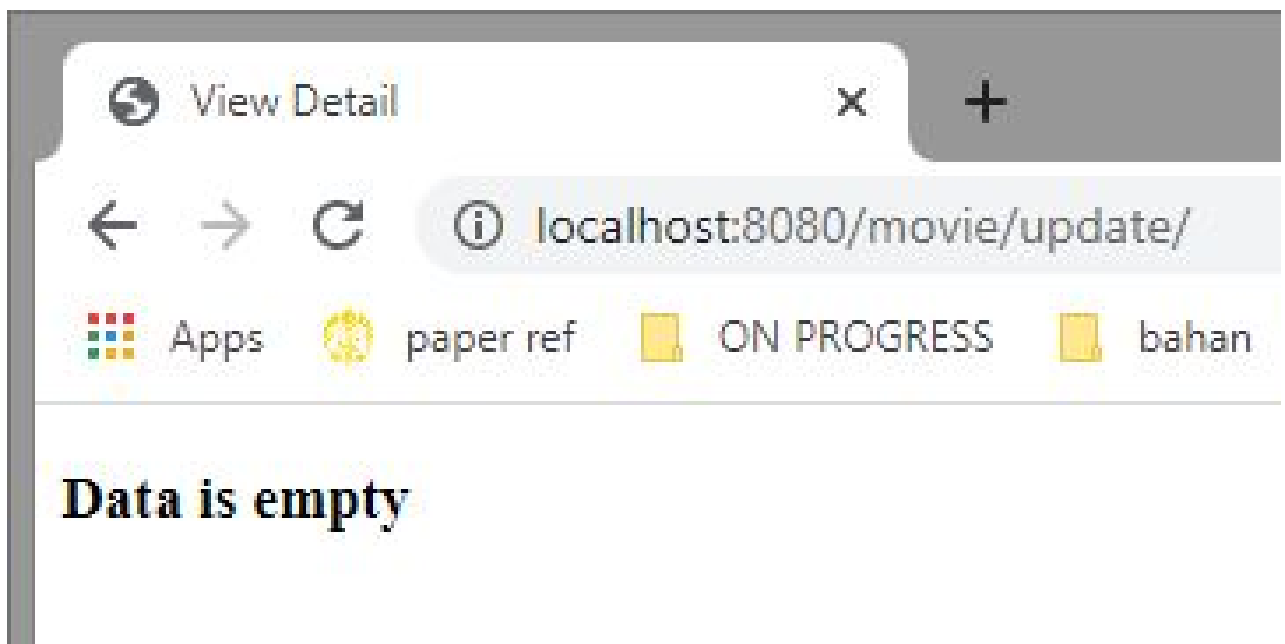
Berikut merupakan perbandingan antara data yang belum diubah dengan data yang telah diubah.

Data	Data
1	1
Insidious	Insidious
Horror	Horror
30000000	30000000
70	60
2	2
ABCD	ABCD
Horror	Horror
30000000	30000000
70	70
3	3
Film ketiga	Film ketiga
Horror	Horror
30000000	30000000
70	70

Apabila id data yang ingin diubah tidak sesuai dengan data yang telah ditambahkan maka akan muncul pesan error “Data not found”



Kemudian apabila tidak memberikan id sebagai parameter maka akan muncul pesan error “Data is empty”.



3. Tambahkan fitur untuk melakukan delete movie berdasarkan id. Misalnya, setelah melakukan perintah nomor 1 dan 2, cobalah untuk melakukan delete data tersebut dengan mengakses halaman: localhost:8080/movie/delete/1. Tampilkan sebuah halaman yang memberikan informasi bahwa data tersebut telah berhasil dihapus. Jika

nomor id tidak diberikan atau tidak ditemukan kembalikan halaman error yang berisi informasi bahwa nomor id kosong atau tidak ditemukan dan proses delete dibatalkan.

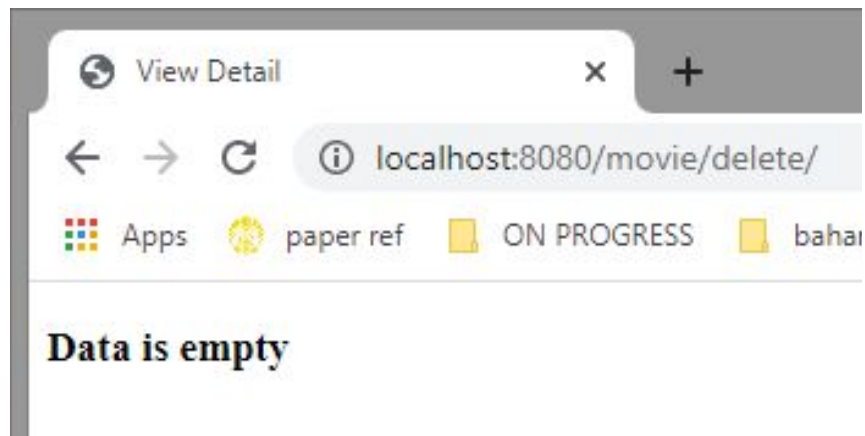
```
MovieService.java X InMemoryMovieService.java MovieCon

src > main > java > com > apap > tu03 > service > MovieService.java > ...
1  package com.apap.tu03.service;
2
3  import java.util.List;
4  import com.apap.tu03.model.MovieModel;
5
6  public interface MovieService {
7
8      List<MovieModel> getMovieList();
9
10     MovieModel getMovieDetail(String id);
11
12     void addMovie(MovieModel movie);
13
14     void deleteMovie(MovieModel movie);
15
16     void updateMovie(MovieModel movie, int duration);
17 }
```

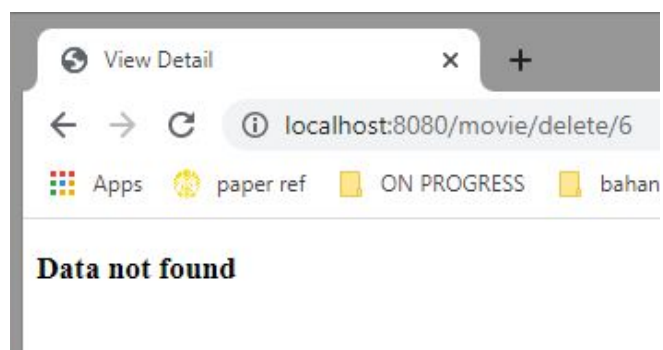
```
@Override
public void deleteMovie(MovieModel movie){
    archiveMovie.remove(movie);
}
```

```
@RequestMapping(value={"/movie/delete", "movie/delete/{id}"})
public String delete(@PathVariable Optional<String> id, Model model){
    if(id.isPresent()){
        MovieModel archive = movieService.getMovieDetail(id.get());
        if(archive == null){
            model.addAttribute("errorMessage", "Data not found");
        }else{
            model.addAttribute("errorMessage", "Data has been removed");
            movieService.deleteMovie(archive);
        }
    }else{
        model.addAttribute("errorMessage", "Data is empty");
    }
    return "errPage";
}
```

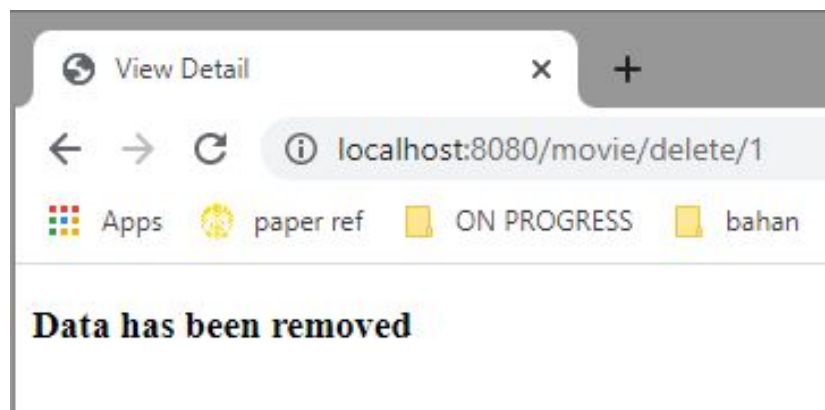
Ketika tidak terdapat parameter pada url, maka pesan error yang muncul adalah “Data is empty”



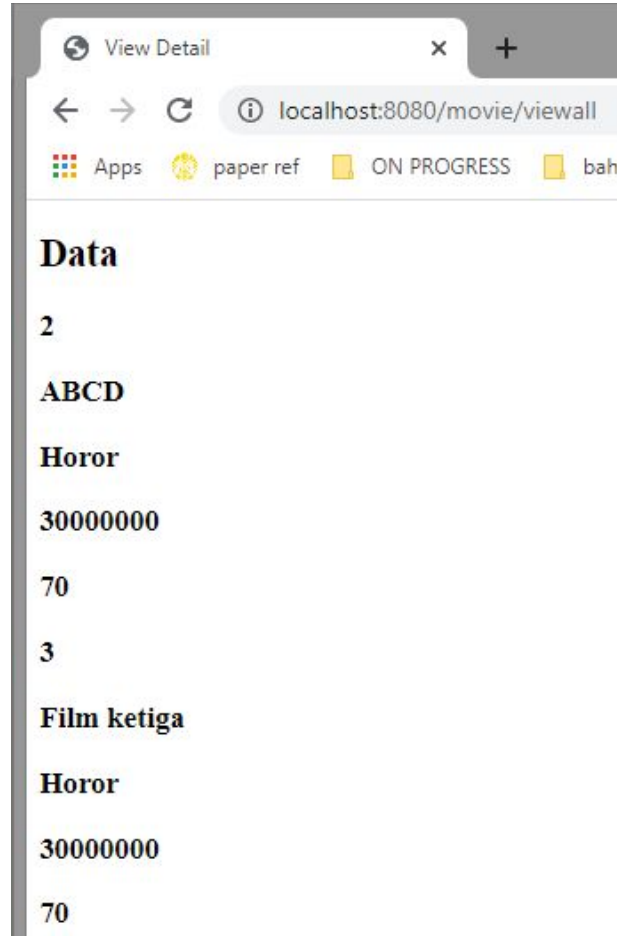
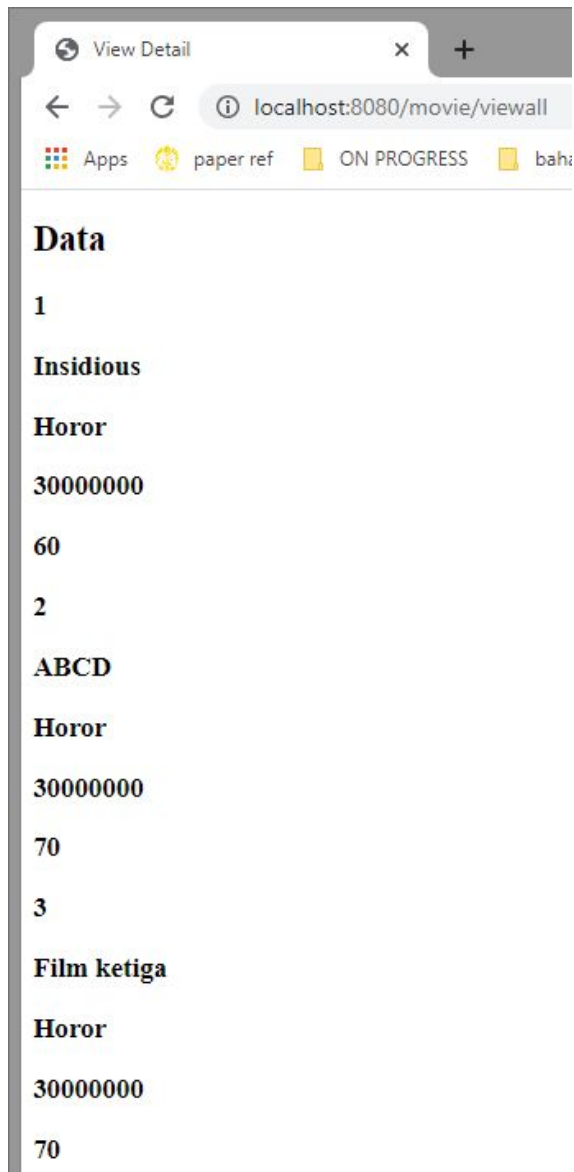
Ketika data yang ingin dihapus tidak sesuai dengan data yang telah ditambahkan sebelumnya maka akan memunculkan pesan error “Data not found”



Ketika data sesuai dengan yang ada maka data tersebut akan dihapus dari List.



Berikut merupakan perbandingan data sebelum dan sesudah dihapus.



Upload Github

```
Riska@RKD01 MINGW64 /e/KULIAH/SI UI/SEMESTER 4/APAP/TUGAS/TUTORIAL/T03/tutorial03 (master)
$ git push -u origin master
Counting objects: 38, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (27/27), done.
Writing objects: 100% (38/38), 54.91 KiB | 2.89 MiB/s, done.
Total 38 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/riskakrndw/tutorial-03.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

[riskakrndw](#) / [tutorial-03](#) Unwatch 1 Star 0 Fork 0

[Code](#) [Issues 0](#) [Pull requests 0](#) [Actions](#) [Projects 0](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

No description, website, or topics provided. [Edit](#)

[Manage topics](#)

1 commit 1 branch 0 packages 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

riskakrndw

 tutorial-03 done Latest commit df08040 2 minutes ago

.mvn/wrapper	tutorial-03 done	2 minutes ago
src	tutorial-03 done	2 minutes ago
.gitignore	tutorial-03 done	2 minutes ago
mvnw	tutorial-03 done	2 minutes ago
mvnw.cmd	tutorial-03 done	2 minutes ago
pom.xml	tutorial-03 done	2 minutes ago

Help people interested in this repository understand your project by adding a README.

[Add a README](#)