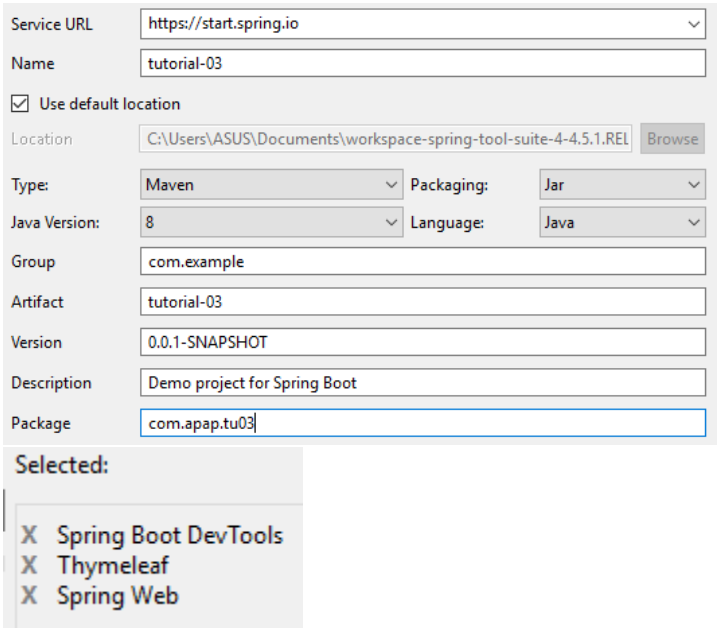
Nama : Izzatunnisa Arwadi

NPM : 1806269726

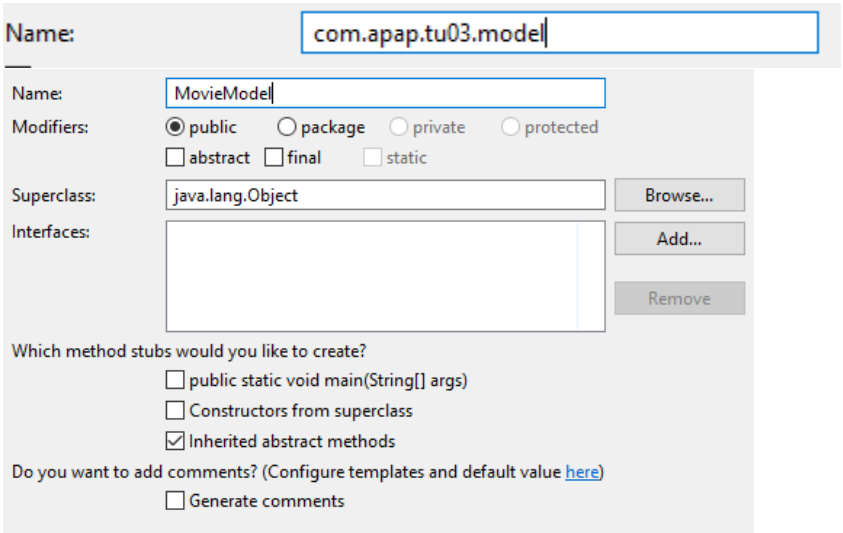
Kelas : Ekstensi

1. Pendahuluan
    - Membuat project baru



Melakukan pembuatan project baru dengan nama tutorial-03, package com.apap.tu03 serta dependencies DevTools, Thymeleaf, dan Web

2. Membuat model
    - Membuat package



Untuk membuat model diawali dengan pembuatan package baru dengan nama com.apap.tu03 model serta dilakukan pembuatan class baru dengan nama MovieModel

- Spesifikasi coding

```
public class MovieModel {
    private String id;
    private String title;
    private String genre;
    private Long budget;
    private Integer duration;
```

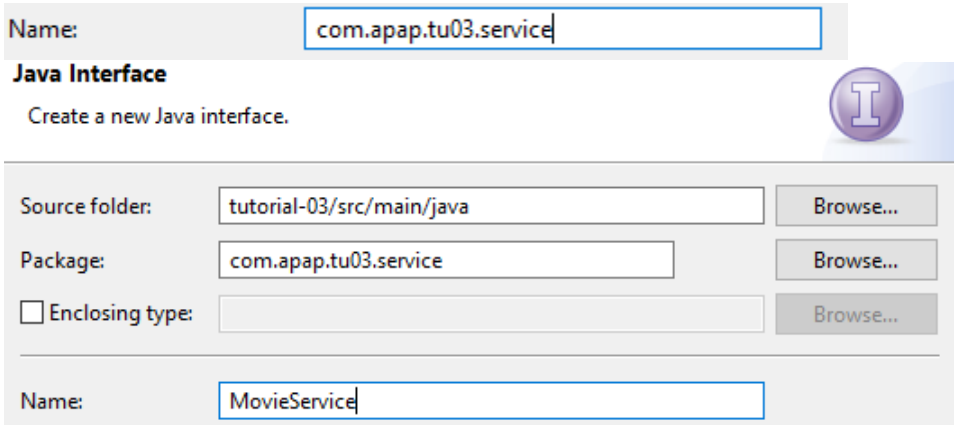
Pada class MovieModel ditambahkan coding seperti di atas.

- Tambahkan method constructor, setter dan getter

```
public MovieModel(String id, String title, String genre, Long budget, Integer duration) {
    this.id = id;
    this.title = title;
    this.genre = genre;
    this.budget = budget;
    this.duration = duration;
}

public String getId() {
    return id;
}

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

public String getTitle() {
    return title;
}

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

public String getGenre() {
    return genre;
}
```

Lalu pada class MovieModel ditambahkan method constructor, setter dan getter.

3. Membuat service
    - Membuat package



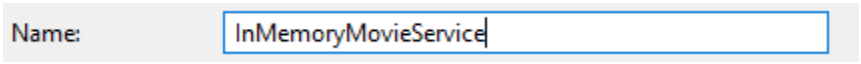
Pembuatan service dilakukan dengan membuat package baru dengan nama com.apap.tu03.service serta pembuatan interface dengan nama MovieService

- Spesifikasi codingan

```java
package com.apap.tu03.service;

import java.util.List;
import com.apap.tu03.model.MovieModel;

public interface MovieService {
    void addMovie(MovieModel movie);
    List<MovieModel> getMovieList();
    MovieModel getMovieDetail(String id);
}
```

Pada interface MovieService tersebut diisikan dengan coding seperti di atas.

- Membuat class dengan spesifikasi codingan




```java
package com.apap.tu03.service;

import java.util.ArrayList;
import java.util.List;

import org.springframework.stereotype.Service;

import com.apap.tu03.model.MovieModel;

@Service
public class InMemoryMovieService implements MovieService{
    private List<MovieModel> archiveMovie;

    public InMemoryMovieService() {
        archiveMovie = new ArrayList<>();
    }

    @Override
    public void addMovie(MovieModel movie) {
        archiveMovie.add(movie);
    }

    @Override
    public List<MovieModel> getMovieList() {
        return archiveMovie;
    }
```

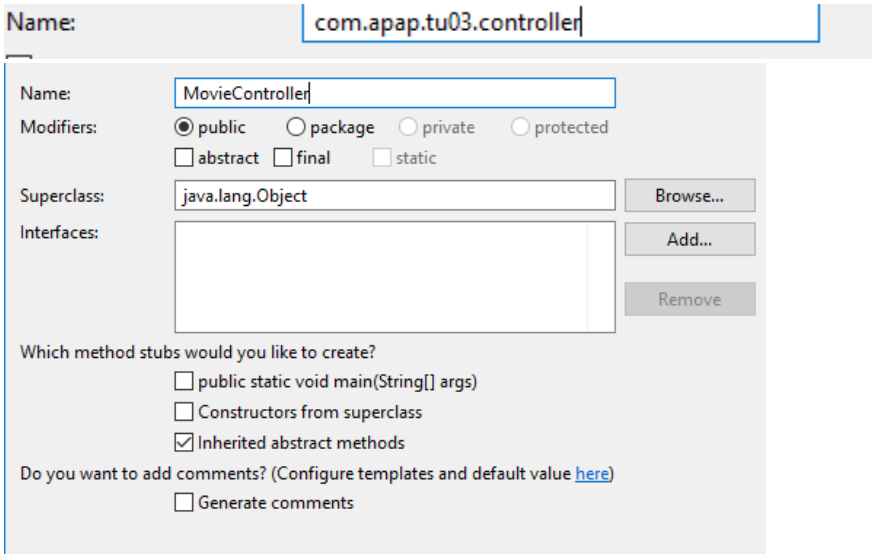
Dilakukan pembuatan class baru dengan nama InMemoryMovieService dengan isi coding seperti di atas.

- Melakukan implementasi method

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

Pada InMemoryMovieService dilakukan implementasi method getMovieDetail yang mengembalikan object MovieModel dengan id tersebut.

4. Membuat Controller dan Method Add
   - Membuat package



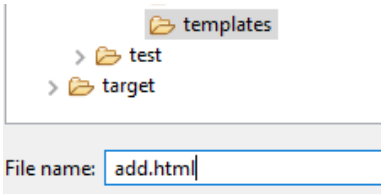
Dilakukan pembuatan package controller yang dinamakan com.apap.tu03.controller dan dibuat class MovieController di dalam package tersebut.

- Membuat class

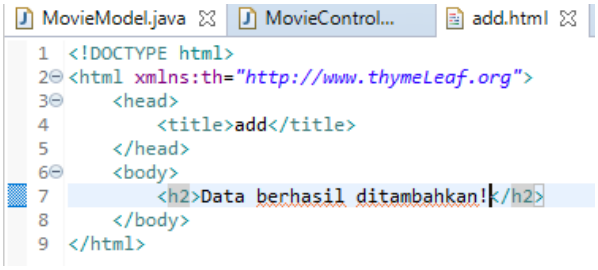
```
package com.apap.tu03.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

import com.apap.tu03.model.MovieModel;
import com.apap.tu03.service.MovieService;

@Controller
public class MovieController {
    @Autowired
    private MovieService movieService;

    @RequestMapping("/movie/add")public String add(@RequestParam(value="id", required=true) String id,
            @RequestParam(value="title", required=true)String title,
            @RequestParam(value="genre", required=true)String genre,
            @RequestParam(value="budget", required=true)Long budget,
            @RequestParam(value="duration", required=true)Integer duration) {
        MovieModel movie=new MovieModel(id, title, genre, budget, duration);
        movieService.addMovie(movie);
        return "add";
    }
}
```

Pada class MovieController diisikan coding seperti di atas.

- Membuat view dengan spesifikasi codingan

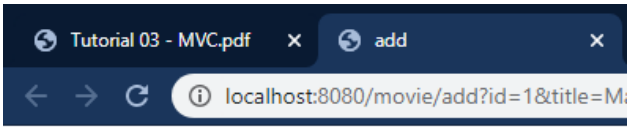


Dilakukan pembuatan view add.html di dalam folder templates.



Pada view add.html diisikan code seperti di atas.

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



Data berhasil ditambahkan!

Ketika dilakukan akses pada http://localhost:8080/movie/add?id=1&title=Marvel&genre=Action&budget=50000000&duration=50 ditampilkan view add.html yakni "Data berhasil ditambahkan!". Hal tersebut terjadi karena parameter id, title, genre, budget dan duration terdapat value. Ketika value ini berhasil di simpan pada masing-masing attribute maka akan ditampilkan view add.html

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

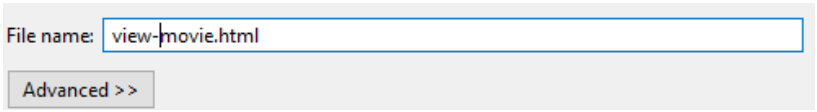
Ketika dilakukan akses http://localhost:8080/movie/add?id=1&title=Marvel&genre=Action&budget

=50000000 ditampilkan whitelabel error page. Hal ini dikarenakan parameter yang dilempar hanya id, title, genre, budget dan tidak terdapat value pada parameter duration. Oleh karena itu ketika attribute dari suatu construct tersebut tidak ada value maka akan ditampilkan error.

5. Membuat method view by ID
   - Penambahan method pada class MovieController

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

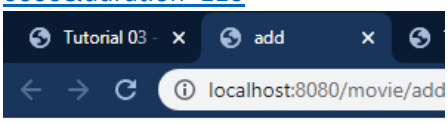
   Dilakukan penambahan method pada class movieController dengan isi codingan seperti di atas.
   - View-movie.html



   Dalam folder templates dibuat file view-movie.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>View Detail</title>
    </head>
    <body>
        <h2>Data</h2>
        <h3 th:text="${movie.id}"></h3>
        <h3 th:text="${movie.title}"></h3>
        <h3 th:text="${movie.genre}"></h3>
        <h3 th:text="${movie.budget}"></h3>
        <h3 th:text="${movie.duration}"></h3>
    </body>
</html>
```

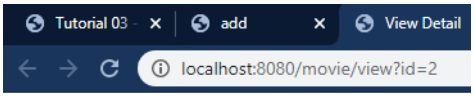
   Dalam file view-movie.html diisikan code seperti di atas.
   - Run project
     - http://localhost:8080/movie/add?id=2&title=Dilan&genre=Romance&budget=10000000&duration=110



Data berhasil ditambahkan!

     Ketika dilakukan akses tersebut ditampilkan view “Data berhasil ditambahkan!”
     - http://localhost:8080/movie/view?id=2

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



Data

2

Dilan

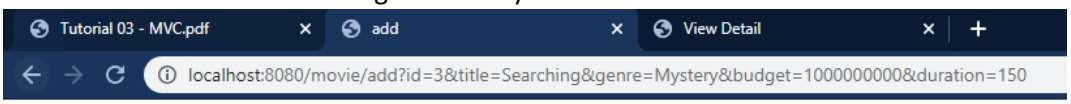
Romance

10000000

110

Pada saat akses http://localhost:8080/movie/view?id=2 ditampilkan id, title, genre, budget serta duration pada movie dengan id 2. Hal ini dikarenakan adanya pemanggilan method view pada controller. Pada method ini akan memanggil attributes pada MovieModel yang sesuai dengan parameter yang telah diinputkan pada saat dilakukan akses.

- Menambahkan data movie dengan id lainnya



Data berhasil ditambahkan!

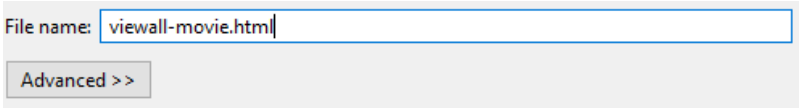
Dilakukan penambahan movie dengan parameter id=3, title=searching, genre=mystery, budget=1000000000 dan duration=150

6. Membuat method view all
   - Penambahan method pada class MovieController

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

Pada class MovieController dilakukan penambahan method viewAll dengan isi coding seperti di atas.

   - Viewall-movie.html

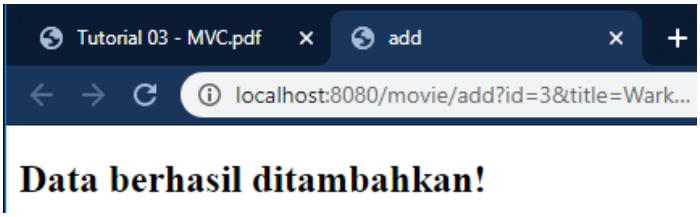


Dilakukan pembuatan file html dalam folder templates yang diberi nama viewall-movie.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>View All</title>
    </head>
    <body>
        <h1>Data</h1>
        <div th:each="movie : ${movies}">
            <h2 th:text="${movie.id}"></h2>
            <h2 th:text="${movie.title}"></h2>
            <h2 th:text="${movie.genre}"></h2>
            <h2 th:text="${movie.budget}"></h2>
            <h2 th:text="${movie.duration}"></h2>
        </div>
    </body>
</html>
```

Pada viewall-movie.html diisikan code seperti di atas.

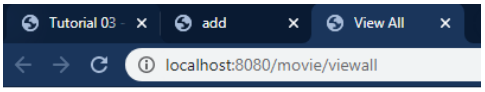
- Run project
  - http://localhost:8080/movie/add?id=3&title=Warkop&genre=Comedy&budget=30000000&duration=70



Ketika dilakukan akses ditampilkan view “Data berhasil ditambahkan!”.

  - http://localhost:8080/movie/viewall

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



Data

3

Warkop

Comedy

30000000

70

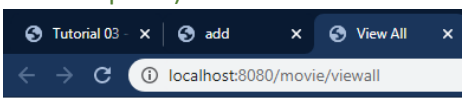
Data movie berhasil ditampilkan. Hal ini dikarenakan pada parameter memenuhi semua attributes pada construct MovieModel. Value yang ada pada parameter disimpan pada masing-masing attribute.

  - Menambahkan id berbeda dan akses http://localhost:8080/movie/viewall

Data berhasil ditambahkan!

Dilakukan penambahan movie dengan parameter id=10, title=Parasite, genre=Thriller, budget= 5000000000 dan duration=110

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



Data

3

Warkop

Comedy

30000000

70

10

Parasite

Thriller

5000000000

110

Semua data movie yang telah di add berhasil ditampilkan. Hal ini dikarenakan ketika dilakukan pemanggilan method add, parameter yang diinputkan dapat disimpan pada masing-masing attribute sehingga ketika dilakukan pemanggilan method viewall dapat ditampilkan semua datanya.

Latihan

1. Ditambahkan method baru yang dinamakan viewById dengan isi codingan seperti di bawah ini

```
@RequestMapping(value= {"/movie/view", "/movie/view/{id}"})
public String viewById(@PathVariable(value="id", required=false) String id, Model model) {
    if(id==null) {
        return "view-kosong";
    }else {
        MovieModel archive = movieService.getMovieDetail(id);
        if(archive==null) {
            return "view-kosong";
        }else {
            model.addAttribute("movie", archive);
            return "view-movie";
        }
    }
}
```

Dilakukan if sebanyak 2 kali. Pada if pertama untuk menentukan apakah terdapat id pada parameter sedangkan pada if kedua untuk mencari apakah id yang diberikan sudah diinputkan atau belum

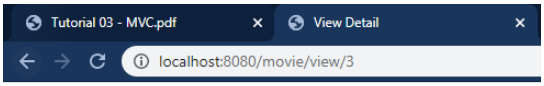
Dikarenakan pada controller terdapat method view yang melakukan requestMapping ke /movie/view sehingga ketika method viewById digunakan akan muncul peringatan ambiguous. Oleh karena itu pada percobaan ini method view dijadikan comment terlebih dahulu

Add data

Data berhasil ditambahkan!

Dilakukan add data dengan id=3

View by id (data ada)



Data

3

Warkop

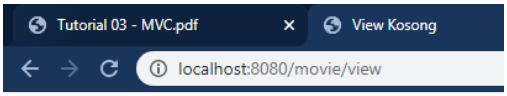
Comedy

30000000

70

Ketika dilakukan akses localhost:8080/movie/view/3 akan ditampilkan data movie dengan id=3

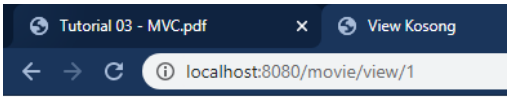
View by id (parameter id tidak ada)



Id kosong atau tidak ditemukan!

Ketika dilakukan akses localhost:8080/movie/view akan ditampilkan peringatan bahwa id kosong atau tidak ditemukan. Hal ini dikarenakan parameter id tidak ada.

View by id (data tidak ditemukan)



Id kosong atau tidak ditemukan!

Ketika dilakukan akses localhost:8080/movie/view/1 akan ditampilkan peringatan bahwa id kosong atau tidak ditemukan. Hal ini dikarenakan id=1 belum diinputkan.

2. Untuk melakukan update diperlukan penambahan method baru pada controller serta pada MovieService dan InMemoryMovieService.

```
void updateDuration(MovieModel movie, Integer duration);
```

Pada MovieService dilakukan deklarasi method tanpa isi

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

Pada InMemoryMovieService terdapat implementasi serta override method updateDuration yang sebelumnya telah dideklarasi pada interface. Method ini akan melakukan pencarian data melalui perulangan serta dilakukan update.

```
@RequestMapping(value= {"/movie/update", "/movie/update/{id}", "/movie/update/{id}/duration/{duration}"})
public String update(@PathVariable(value="id", required=false) String id, @PathVariable(value="duration",
    if(id==null && duration==null) {
        return "view-kosong-update-gagal";
    }else {
        MovieModel archive = movieService.getMovieDetail(id);
        if(archive==null) {
            return "view-kosong-update-gagal";
        }else {
            model.addAttribute("movie", archive);
            movieService.updateDuration(archive, duration);
            return "update";
        }
    }
}
```

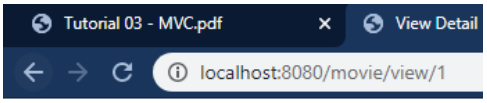
Pada controller ditambahkan method update. Didalamnya terdapat 2 if, pada if pertama untuk menentukan apakah id dan duration pada parameter memiliki value null atau tidak sedangkan pada if kedua untuk menentukan apakah data movie tersebut ada atau tidak. Ketika data movie ada, maka dilakukan m=pemanggilan method updateDuration.

Add data

Data berhasil ditambahkan!

Dilakukan add data dengan id=1

View by id



Data

1

Marvel

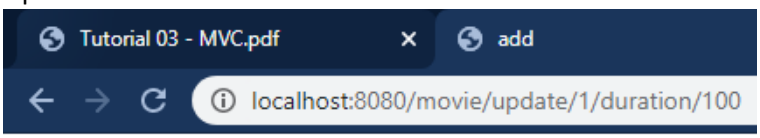
Action

50000000

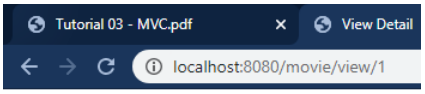
50

Ketika dilakukan akses localhost:8080/movie/view/1 akan ditampilkan data movie dengan id=1

Update



Data berhasil diubah!



Data

1

Marvel

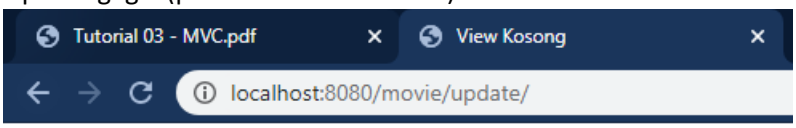
Action

50000000

100

Ketika dilakukan akses localhost:8080/movie/update/1/duration/100 ditampilkan notifikasi bahwa data berhasil diubah. Ketika dilakukan akses localhost:8080/movie/view/1 dapat terlihat bahwa durasi telah berubah.

Update gagal (parameter id tidak ada)

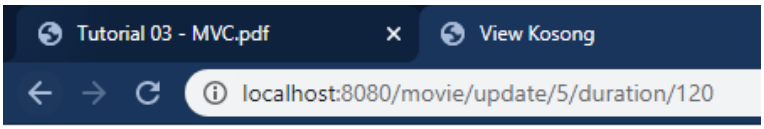


Id kosong atau tidak ditemukan!

Update Gagal

Ketika id parameter null, maka akan ditampilkan peringatan update gagal.

Update gagal (id tidak ditemukan)



Id kosong atau tidak ditemukan!

Update Gagal

Ketika id yang dicari tidak ada, maka akan ditampilkan peringatan update gagal.

3. Untuk melakukan delete diperlukan penambahan method baru pada controller serta pada MovieService dan InMemoryMovieService.

```
void deleteMovie(MovieModel movie);
```

Dilakukan pembuatan method tanpa isi pada interface MovieService.

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

Pada InMemoryMovieService dilakukan pembuatan method baru dengan menggunakan perintah remove.

```
@RequestMapping(value= {"/movie/delete", "/movie/delete/{id}"})
public String delete(@PathVariable(value="id", required=false) String id, Model model) {
    if(id==null) {
        return "view-kosong-delete-gagal";
    }else {
        MovieModel archive = movieService.getMovieDetail(id);
        if(archive==null) {
            return "view-kosong-delete-gagal";
        }else {
            model.addAttribute("movie", archive);
            movieService.deleteMovie(archive);
            return "delete";
        }
    }
}
```

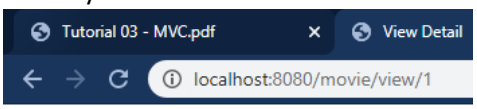
Pada controller ditambahkan method delete. Didalamnya terdapat 2 if, pada if pertama untuk menentukan apakah id pada parameter memiliki value null atau tidak sedangkan pada if kedua untuk menentukan apakah data movie tersebut ada atau tidak. Ketika data tersebut ada akan dilakukan pemanggilan method deleteMovie.

Add data

Data berhasil ditambahkan!

Dilakukan add data dengan id=1

View by id



Data

1

Marvel

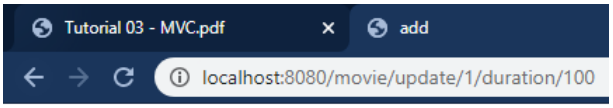
Action

50000000

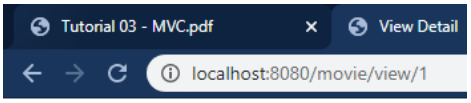
50

Ketika dilakukan akses localhost:8080/movie/view/1 akan ditampilkan data movie dengan id=1

Update



Data berhasil diubah!



Data

1

Marvel

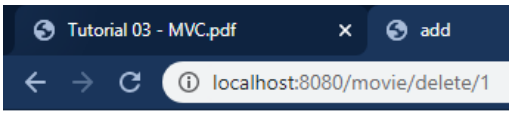
Action

50000000

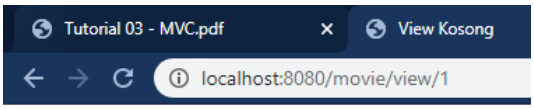
100

Ketika dilakukan akses localhost:8080/movie/update/1/duration/100 ditampilkan notifikasi bahwa data berhasil diubah. Lalu, ketika dilakukan akses localhost:8080/movie/view/1 dapat terlihat bahwa durasi telah berubah.

Delete



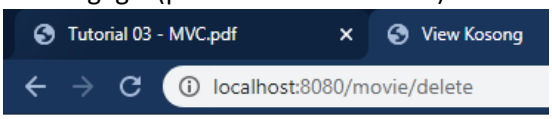
Delete berhasil!



Id kosong atau tidak ditemukan!

Ketika dilakukan akses ocalhost:8080/movie/delete/1 maka data dengan id=1 telah terhapus dan muncul notifikasi bahwa delete berhasil. Lalu, ketika dilakukan akses localhost:8080/movie/view/1 dapat terlihat bahwa muncul notifikasi id kosong atau tidak ditemukan.

Delete gagal (parameter id tidak ada)

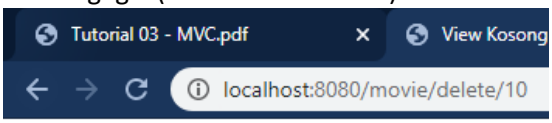


Id kosong atau tidak ditemukan!

Delete Gagal

Ketika id parameter null, maka akan ditampilkan peringatan delete gagal.

Delete gagal (id tidak ditemukan)



Id kosong atau tidak ditemukan!

Delete Gagal

Ketika id yang dicari tidak ditemukan, maka akan ditampilkan peringatan delete gagal.