

Ringkasan

Pada tutorial kali ini, dipelajari bagaimana cara mendesain presentation layer pada sebuah web aplikasi. Proses desain dikerjakan dengan menggunakan thymeleaf, bootstrap dan datatable. Adapun hal-hal yang dapat dipelajari pada kesempatan ini adalah bagaimana penggunaan looping pada thymeleaf, penggunaan condition evaluation pada thymeleaf, penggunaan static file (fragments) dan mengimplementasikan penggunaan error handler seperti http 404, http 500 dan sebagainya. Kemudian, dipelajari juga bagaimana cara menginstalasi dan menggunakan bootstrap maupun datatable.

Jawaban dari Pertanyaan

1. Apakah value yang dihasilkan dari `${iterationStatus.odd}`?

Jawaban:

Value yang dihasilkan adalah menambahkan attribute class dengan nama “odd” pada tag HTML `<div>` ketika nilai iterasi dari list student bernilai ganjil, hasilnya dapat dilihat seperti Gambar 1 berikut (sebagai catatan java menggunakan *zero-based indexing*, sehingga *record* urutan ke-2 dan kelipatannya akan dianggap sebagai iterasi ganjil).



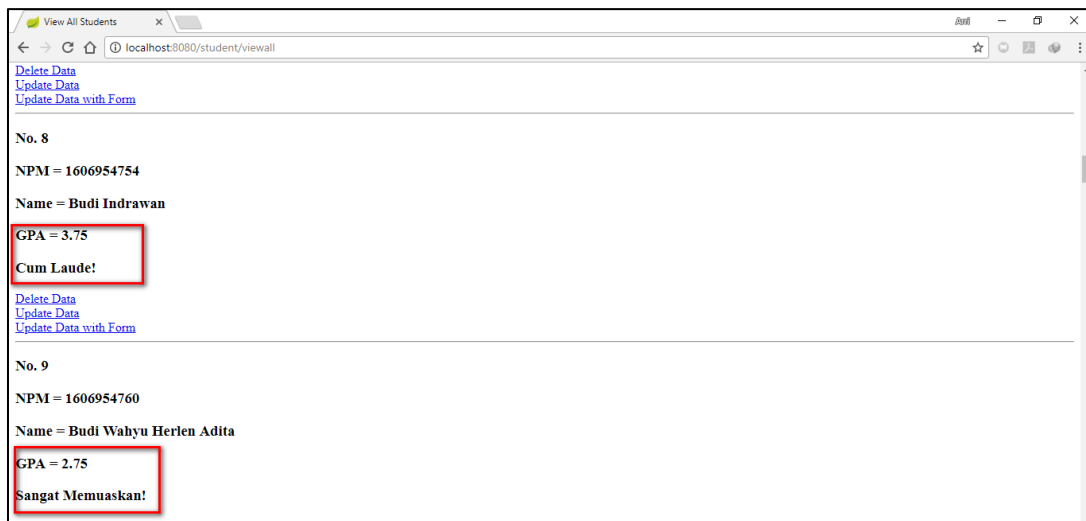
Gambar 1. Page source viewall.html

2. Mengapa condition di dalam **th:unless** sama dengan di dalam **th:if** ? Jelaskan jawaban Anda.

Jawaban:

Kondisi tersebut sama, karena nilai yang dihasilkan **th:unless** merupakan kebalikan dari kondisi yang diinginkan (dalam kasus di atas, berarti nilai yang memenuhi adalah nilai gpa yang **tidak lebih dan sama dengan 3.49** dinyatakan sebagai sangat memuaskan) atau secara notasi **th:if** dapat dituliskan seperti berikut: **th:if="\${not student.gpa}>=3.49}"**. Sehingga, kondisi yang sama tersebut hanya digunakan untuk menampilkan status dari

nilai gpa student jika lebih besar dan sama dengan 3.49 dinyatakan “Cum Laude!” dan sebaliknya dinyatakan “Sangat Memuaskan!” (Lihat Gambar 2).

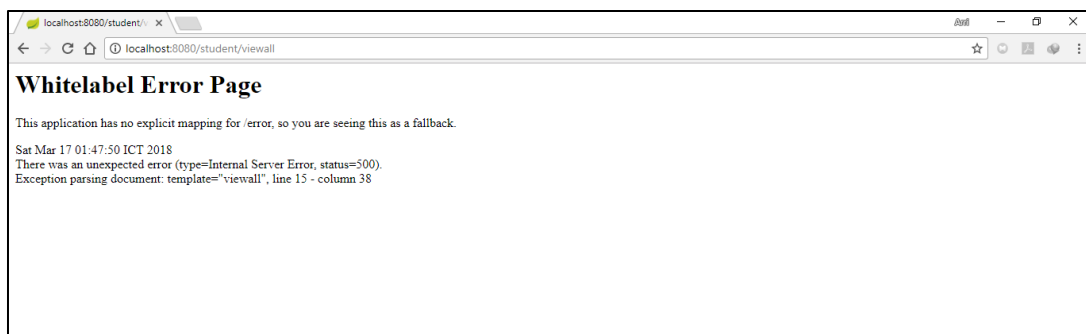


Gambar 2. Page viewall.html

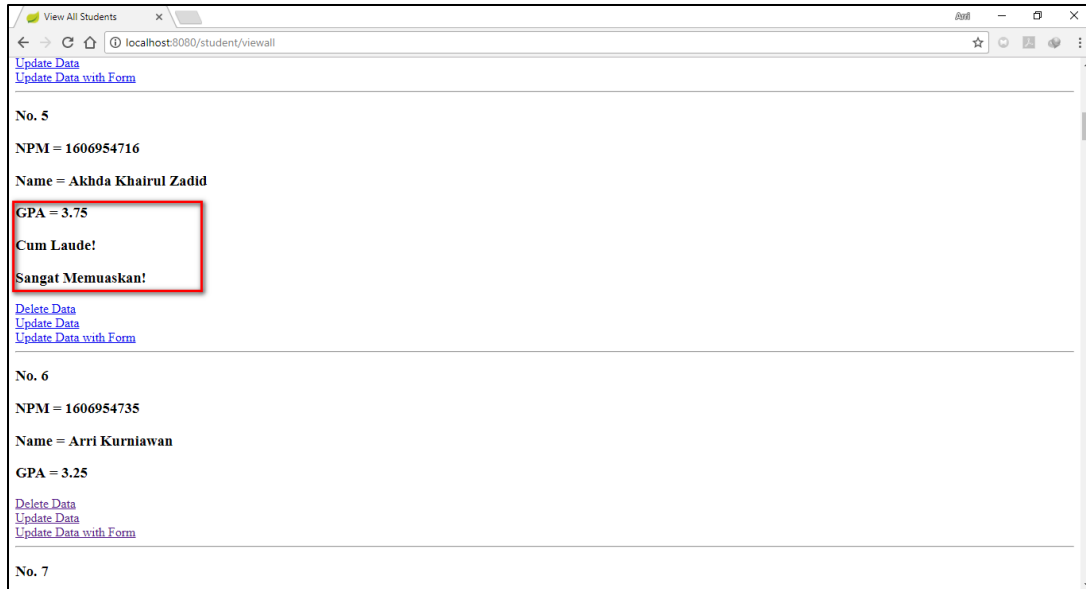
3. Coba ubah condition pada **th:unless=\${student.gpa}>=3.49}** menjadi **th:unless=\${student.gpa}<=3.48}**, lalu run kembali aplikasi Anda dan mengakses halaman yang sama. Apakah terjadi error? Jika iya, kenapa? Jelaskan alasan Anda. Jika tidak error, apakah hasilnya sesuai?

Jawaban:

Iya terjadi error, sebab ada error parsing seperti terlihat pada Gambar 3. Error parsing tersebut terjadi dikarenakan oleh tanda “<” pada bagian **th:unless=\${student.gpa}<=3.48}**, di mana browser membaca tersebut sebagai awal baru dari sebuah tag HTML sehingga tag HTML yang sebelumnya digunakan tidak dapat membaca closing tag-nya. Oleh karenanya, jika ingin menggunakan tanda “<” tersebut pada syntax thymeleaf harus menggunakan encode seperti “<”. Adapun hasil yang didapatkan setelah menggunakan encode tersebut dapat dilihat pada Gambar 4. Di mana pada Gambar 4, hasil yang ditampilkan memang memenuhi kondisi yang ada akan tetapi hasilnya tidak sesuai dengan data yang diharapkan.



Gambar 3. Error Parsing Page viewall.html



Gambar 4. Page viewall.html (wrong condition)

4. Tuliskan cara lain Anda dengan penulisan conditional expression yang berbeda untuk mendapatkan hasil yang sama seperti gambar di atas. **Hint: Gunakan ternary operator.**

Jawaban:

Adapun cara lain penulisan conditional expression adalah dengan menggunakan ternary operator seperti potongan code yang dapat di lihat pada Gambar 5.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  <head>
    <title>View All Students</title>
  </head>
  <body>
    <h1>All Students</h1>

    <div th:each="student, iterationStatus: ${students}" th:class="${iterationStatus.odd} ? 'odd'"
      <h3 th:text="'No. ' + ${iterationStatus.count}">No. 1</h3>
      <h3 th:text="'NPM = ' + ${student.npm}">Student NPM</h3>
      <h3 th:text="'Name = ' + ${student.name}">Student Name</h3>
      <h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>
      <h3 th:text="${student.gpa}=3.49 ? 'Cum Laude!' : 'Sangat Memuaskan!'">Cum Laude!</h3>
      <a th:href="/student/delete/" + ${student.npm}">Delete Data</a><br />
      <a th:href="/student/update/" + ${student.npm}">Update Data</a><br />
      <a th:href="/student/updateForm/" + ${student.npm}">Update Data with Form</a><br />
    </div>
  </body>
</html>
```

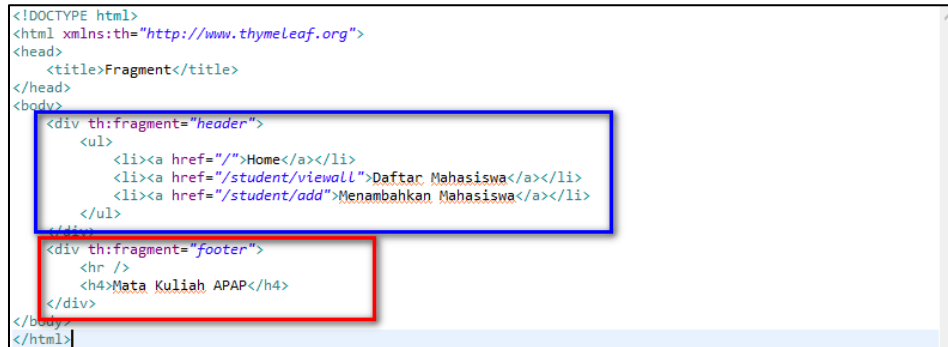
Gambar 5. Penggunaan Ternary Operator

Seperti yang terlihat pada Gambar 5, statement setelah tanda "?" merupakan statement yang akan ditampilkan karena memenuhi kondisi, sedangkan setelah tanda ":" merupakan statement yang akan ditampilkan karena tidak memenuhi kondisi.

5. Apa yang dimaksud dengan **th:replace="fragments/fragment :: header"** dan **th:replace="fragments/fragment :: footer"** pada file index.html yang Anda buat?

Jawaban:

Maksud dari kedua statement tersebut adalah menggantikan bagian pada index.html dengan tampilan yang telah dibuat pada fragment.html sesuai dengan fragment masing-masing (lihat Gambar 6). Di mana **th:replace="fragments/fragment :: header"** digantikan oleh bagian pada kotak biru, sedangkan **th:replace="fragments/fragment :: footer"** digantikan oleh kotak merah.



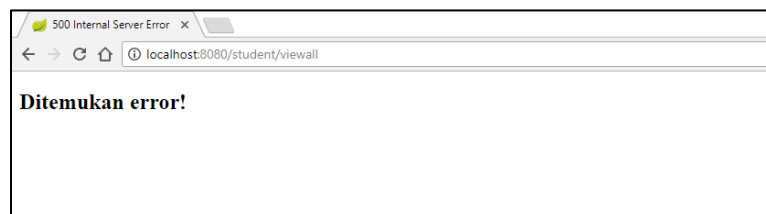
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <title>Fragment</title>
</head>
<body>
  <div th:fragment="header">
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/student/viewall">Daftar Mahasiswa</a></li>
      <li><a href="/student/add">Menambahkan Mahasiswa</a></li>
    </ul>
  </div>
  <div th:fragment="footer">
    <hr />
    <h4>Mata Kuliah APAP</h4>
  </div>
</body>
</html>
```

Gambar 6. Page fragment.html

6. Apa handler dengan metode ini juga dapat berlaku bagi error lain seperti **error 500: internal server error**?

Jawaban:

Iya, handler tersebut dapat berlaku bagi error http lainnya seperti error 500. Gambar 7 menunjukkan tampilan error 500 setelah dilakukan perubahan source code untuk bagian viewall dengan beberapa statement agar tampilan tersebut error.



Gambar 7. Error Handler HTTP 500

Latihan – Penggunaan Datatables

Pada Gambar 8, dapat dilihat perubahan yang dilakukan terhadap viewall.html. Untuk mengimplementasikan penggunaan datatables.js, pertama-tama harus dilakukan pemanggilan resource javascript maupun css-nya (dapat dilihat pada kotak merah di Gambar 9). Kemudian, selanjutnya menambahkan fungsi javascript seperti pada kotak biru di Gambar 9. Adapun hasil dari pengimplementasian ini dapat dilihat pada Gambar 10.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head th:replace="fragments/fragment :: assets" />
<body>
<div th:replace="fragments/fragment :: header"></div>
<h1>All Students</h1>
<table id="studentTable" class="display" style="width:100%">
<thead>
<tr>
<th>No.</th>
<th>NPM</th>
<th>Name</th>
<th>GPA</th>
<th>Status</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr th:each="student, iterationStatus: ${students}">
<td th:text="${iterationStatus.count}"></td>
<td th:text="${student.npm}"></td>
<td th:text="${student.name}"></td>
<td th:text="${student.gpa}"></td>
<td th:text="${student.gpa}>3.49 ? 'Cum Laude!' : 'Sangat Memuaskan!'"></td>
<td><a th:href="/student/delete/" + ${student.npm}>Delete Data</a></td>
<td><a th:href="/student/update/" + ${student.npm}>Update Data</a></td>
</tr>
</tbody>
</table>
<div th:replace="fragments/fragment :: footer"></div>
</body>
</html>
```

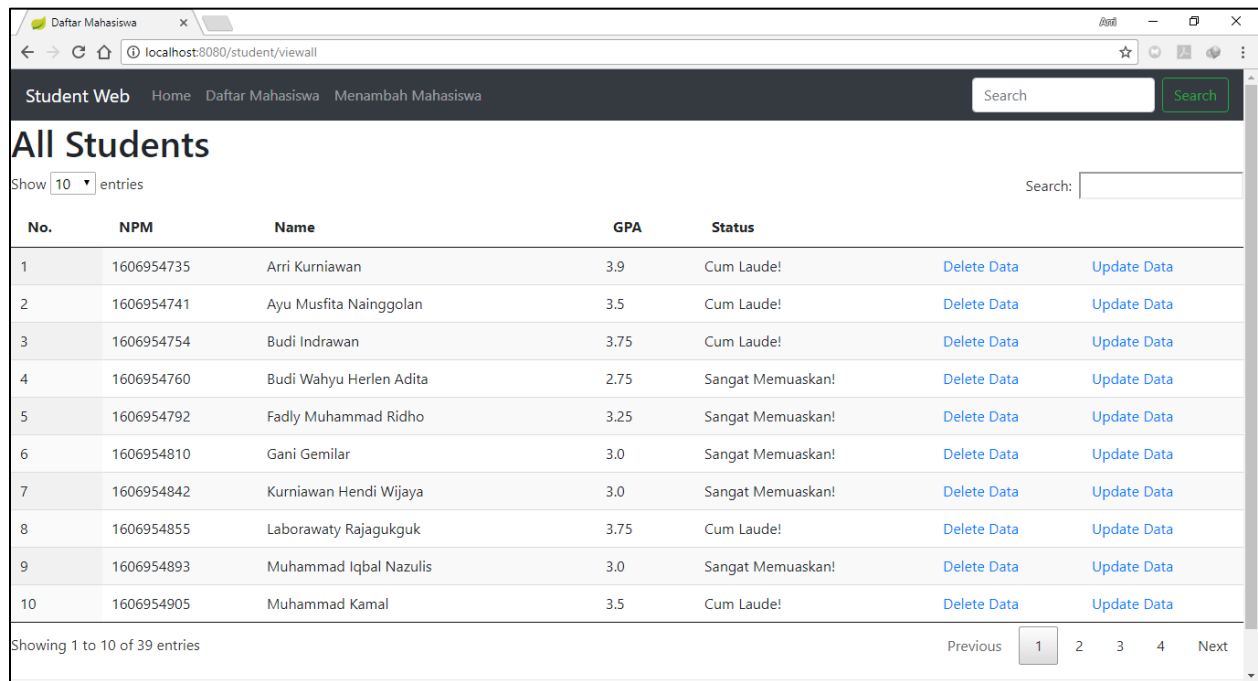
Gambar 8. Source code viewall.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head th:fragment="assets">
<title th:text="${title}"></title>
<link rel="stylesheet" href="/css/bootstrap.min.css" />
<link rel="stylesheet" href="/css/datatables.min.css" />
<script type="text/javascript" src="/js/jquery-3.2.1.min.js"></script>
<script type="text/javascript" src="/js/bootstrap.min.js"></script>
<script type="text/javascript" src="/js/datatables.min.js"></script>
<script type="text/javascript">
$(document).ready(function() {
$('#studentTable').DataTable();

$('.navbar-nav li').click(function(){
$('.navbar-nav li').removeClass('active');
$(this).addClass('active');
});
});
</script>
</head>
<body>
<div th:fragment="header">
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
<a class="navbar-brand" href="#">Student Web</a>
<button class="navbar-toggler" type="button"
data-toggle="collapse" data-target="#navbarSupportedContent"
aria-controls="navbarSupportedContent" aria-expanded="false"
aria-label="Toggle navigation">
<span class="navbar-toggler-icon"></span>
</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">
<ul class="navbar-nav mr-auto">
<li class="nav-item">
<a class="nav-link" href="/">Home</a>
</li>
<li class="nav-item">
<a class="nav-link" href="/student/viewall">Daftar Mahasiswa</a>
</li>
</ul>
</div>
</div>
</div>
```

Gambar 9. Pengimplementasian datatables.js



No.	NPM	Name	GPA	Status		
1	1606954735	Arri Kurniawan	3.9	Cum Laude!	Delete Data	Update Data
2	1606954741	Ayu Musfita Nainggolan	3.5	Cum Laude!	Delete Data	Update Data
3	1606954754	Budi Indrawan	3.75	Cum Laude!	Delete Data	Update Data
4	1606954760	Budi Wahyu Herlen Adita	2.75	Sangat Memuaskan!	Delete Data	Update Data
5	1606954792	Fadly Muhammad Ridho	3.25	Sangat Memuaskan!	Delete Data	Update Data
6	1606954810	Gani Gemilar	3.0	Sangat Memuaskan!	Delete Data	Update Data
7	1606954842	Kurniawan Hendi Wijaya	3.0	Sangat Memuaskan!	Delete Data	Update Data
8	1606954855	Laborawaty Rajagukguk	3.75	Cum Laude!	Delete Data	Update Data
9	1606954893	Muhammad Iqbal Nazulis	3.0	Sangat Memuaskan!	Delete Data	Update Data
10	1606954905	Muhammad Kamal	3.5	Cum Laude!	Delete Data	Update Data

Gambar 10. Hasil dari pengimplentasian datatables.js

Latihan – Penggunaan Fragments (tampilan dinamis)

Gambar 11 menunjukkan fragment.html yang telah dirubah untuk keperluan latihan ini. Untuk membuat dinamis perubahan title dapat dilihat pada bagian dengan kotak merah di Gambar 11. Nilai “**`\${title}`**” didapatkan dari **StudentController** dengan menyisipkan sebuah attribute pada masing-masing method yang dipanggil seperti pada bagian kotak biru pada Gambar 12.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head th:fragment="header">
    <title th:text="${title}"></title>
    <link rel="stylesheet" href="/css/bootstrap.min.css" />
    <link rel="stylesheet" href="/css/datatables.min.css" />
    <script type="text/javascript" src="/js/jquery-3.2.1.min.js"></script>
    <script type="text/javascript" src="/js/bootstrap.min.js"></script>
    <script type="text/javascript" src="/js/datatables.min.js"></script>
    <script type="text/javascript">
        $(document).ready(function() {
            $('#studentTable').DataTable();

            $('.navbar-nav li').click(function(){
                $('.navbar-nav li').removeClass('active');
                $(this).addClass('active');
            });
        });
    </script>
</head>
<body>
    <div th:fragment="header">
        <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
            <a class="navbar-brand" href="#">Student Web</a>
            <button class="navbar-toggler" type="button"
                data-toggle="collapse" data-target="#navbarSupportedContent"
                aria-controls="navbarSupportedContent" aria-expanded="false"
                aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarSupportedContent">
                <ul class="navbar-nav mr-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="/">Home</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/student/viewall">Daftar Mahasiswa</a>
                    </li>
                </ul>
            </div>
        </nav>
    </div>
</body>
</html>
```

Gambar 11. Source Code fragment.html

```
package com.example.controller;

import java.util.List;

@Controller
public class StudentController
{
    @Autowired
    StudentService studentDAO;

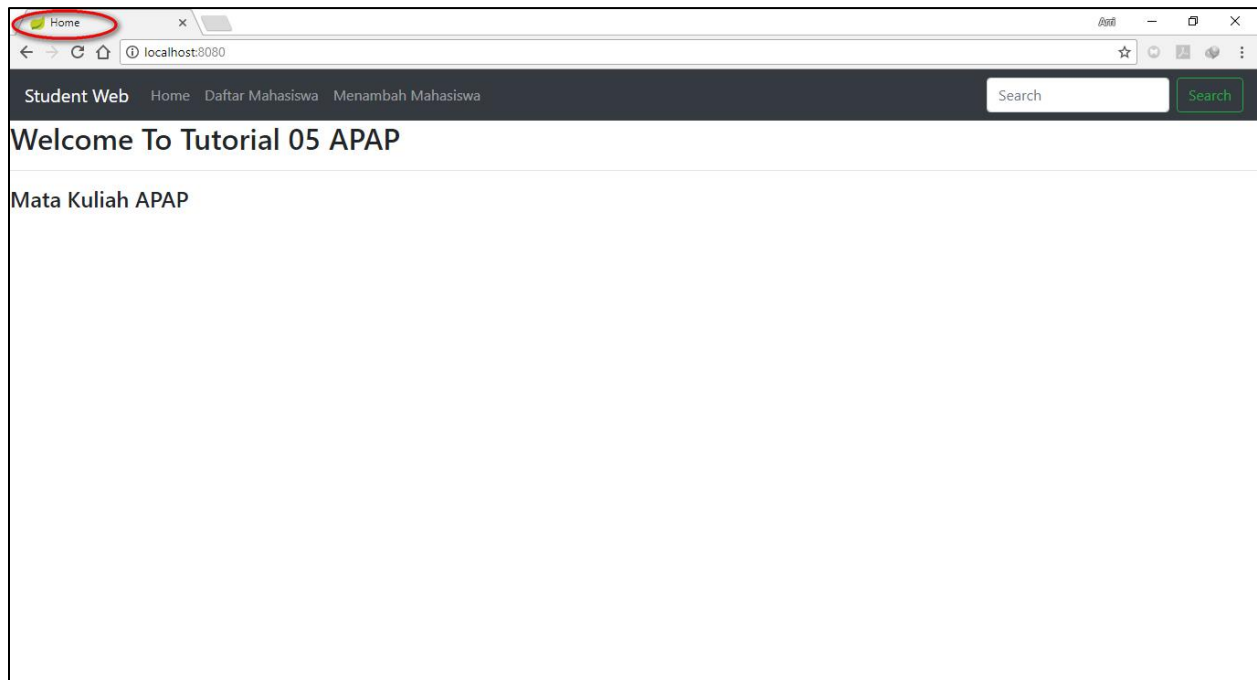
    @RequestMapping("/")
    public String index (Model model)
    {
        model.addAttribute("title", "Home");
        return "index";
    }

    @RequestMapping("/student/add")
    public String add (Model model)
    {
        model.addAttribute("title", "Menambah Mahasiswa");
        return "form-add";
    }

    @RequestMapping("/student/add/submit")
    public String addSubmit (Model model,
        @RequestParam(value = "npm", required = false) String npm,
        @RequestParam(value = "name", required = false) String name,
        @RequestParam(value = "gpa", required = false) double gpa)
    {
        StudentModel student = new StudentModel (npm, name, gpa);
        studentDAO.addStudent (student);
        model.addAttribute("title", "Sukses Menambah Mahasiswa");
        return "success-add";
    }
}
```

Gambar 12. Source Code StudentController

Sehingga , hasilnya dapat dilihat pada Gambar 13.



Gambar 13. Page index.html