1. ${iterationStatus.odd} value yang dihasilkan merupakaan boolean bernilai true jika value dari iterationStatus bernilai ganjil
2. Karena th:unless bekerja seperti else di if. Sehingga jika nilainya tidak seperti di kondisi th:unless, maka akan bernilai true. Jadi pada th:unless jika gpa bernilai <=3.48, maka dia akan bernilai true dan menghasilkan tampilan “Sangat Memuaskan!”
3. Terjadi error pada program karena pada dasarnya akan terjadi logical error. Karena nilai true pada th:if dan th:unless akan menjadi sama. Sehingga muncul error exception parsing pada kasus tersebut. Karena pada prinsipnya th:unless merupakan inverse attribute dari th:if
4. th:text="${student.gpa>=3.49} ? ‘Cum Laude!’ : ‘Sangat Memuaskan!’"
5. th:replace="fragments/fragment :: header" nantinya akan menggantikan/import kode sesuai dengan yang ada pada fragment.html dan bernama header. Sedangkan th:replace="fragments/fragment :: footer" juga akan menggantikan/import kode yang ada pada fragment.html dan bernama footer.
6. Bisa, handler itu dapat digunakan untuk 500 Internal Server Error. Tetapi harus membuat page htmlnya kembali dengan nama 500.html sesuai dengan nomor errornya.

LATIHAN

1. Pertama saya download semua file yang dibutuhkan untuk dataTables lalu melakukan instalasi pada viewall.html. Saya juga membuat file template.js agar dataTables dapat berjalan.

```
<script src="/js/jquery-3.2.1.min.js"></script>
<script src="/js/datatables.min.js"></script>
<script src="/js/template.js"></script>
```

```
$(document).ready(function () {
    $('#example').DataTable(
        {
            "paging": false,
            "info": false,
            searching: false
        });
});
```

Selanjutnya saya memperbaiki kode agar tampilannya dapat terdistribusi dalam table.

```
<table id="example" class="display" cellspacing="0" width="100%">
    <thead>
        <tr>
            <td>No</td>
            <td>NPM</td>
            <td>Name</td>
            <td>GPA</td>
            <td>Cum laude</td>
            <td>Delete</td>
            <td>Update</td>
        </tr>
    </thead>

    <tbody>
        <tr th:each="student,iterationStatus: ${students}">
            <td th:text="${iterationStatus.count}">No. 1</td>
            <td th:text="${student.npm}">Student NPM</td>
            <td th:text="${student.name}">Student Name</td>
            <td th:text="${student.gpa}">Student GPA</td>
            <td th:if="${student.gpa>=3.49}">Cum Laude!</td>
            <td th:unless="${student.gpa>=3.49}">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>
```

Sehingga tampilannya akan berubah menjadi seperti ini

All Students

No	NPM	Name	GPA	Cum laude	Delete	Update
1	123	Chanek	4.0	Cum Laude!	Delete Data	Update Data
2	124	Chanek Jr	3.0	Sangat Memuaskan!	Delete Data	Update Data

2. Pertama saya membuat tag head pada fragment.html agar dapat menjadi dynamic saat digunakan pada bagiann title dan instalasi css serta jsnya dengan kode sebagai berikut

```
<head th:fragment="header">
    <title th:text="${page_title}">Title</title>
    <script th:if="${page == 'viewall'}" src="/js/jquery-3.2.1.min.js"></script>
    <script th:if="${page == 'viewall'}" src="/js/datatables.min.js"></script>
    <script th:if="${page == 'viewall'}" src="/js/template.js"></script>
    <link rel="stylesheet" href="/css/bootstrap.min.css" />
    <link rel="stylesheet" href="/css/datatables.min.css" />
</head>
```

Kemudia dengan konsep dynamic pada bagian title yang seperti itu, maka disetiap return mapping akan saya tambahkan atribut model dengan nama page_title yang nilainya dapat saya tentukan sesuai yang saya mau, contohnya seperti pada gambar dibawah ini

```
model.addAttribute( attributeName: "page_title", attributeValue: "Mata Kuliah " + course.getName() + " - " + id_course);
return "view-course";
```

Selanjutnya saya melakukan replace head pada setiap html view yang akan tampil

```
<head th:replace="fragments/fragment :: header"></head>
```

Lesson Learned:
Pada tutorial kali ini saya mempelajari terkait pemanfaatan thymeleaf agar mempermudah dalam tampilan pada view. Selain itu saya juga mempelajari terkait fragments untuk mempermudah pengaturan tampilan agar di setiap viewnya tidak perlu dibuat kode yang mengulang dan sama. Selain itu saya juga mempelajari terkait handle error sehingga tampilan error dapat di atur sesuai dengan yang diinginkan.