Tutorial 6 Unit Testing dan Load Testing Web App Sederhana

Nama : Esther Theresia Sihotang
NPM : 1506797242

Latihan Unit Testing

1. Tambahkan *code* **selectAllStudent** pada StudentServiceDatabase. Jelaskan apa yang terjadi pada method tersebut;

Jawab : pada method selectAllStudent terdapat 3(tiga) segmen yaitu Given, When dan Then yang akan dilakukan pada setiap *testing*.

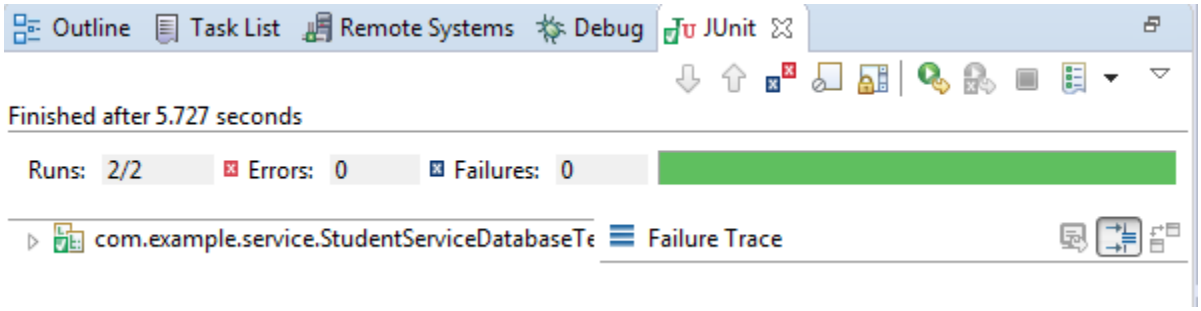
- Pada **Given**, pengguna menginisialisasi objek studentModels pada *arraylist* kemudian BDDMockito melakukan proses sesuai *method* **selectAllStudent** dengan hasil *return value* sesuai dengan willReturn(studentModels).
- Pada **When**, akan dilakukan pengetesan terhadap method **selectAllStudent** dengan memasukkan semua objek *student* yang ada pada arraylist apakah method **selectAllStudent** tetap berjalan.
- Pada **Then**, akan dilakukan pengecekan terhadap objek baru tersebut. Objek baru dicek jika null, cek kalau ukuran sama, cek kalau konten sama atau sesuai dengan objek check yang telah disiapkan sebelumnya.

2. Tambahkan *code* **addStudent** pada StudentServiceDatabase. Jelaskan apa yang terjadi pada method tersebut

Jawab : pada method addStudent terdapat 3(tiga) segmen yaitu Given, When dan Then yang akan dilakukan pada setiap *testing*.

- Pada **Given**, pengguna menginisialisasi objek studentModels kemudian BDDMockito melakukan proses sesuai *method* **addStudent** dengan hasil *return value* sesuai dengan willReturn(studentModels).
- Pada **When**, akan dilakukan pengetesan terhadap method **addStudent** dengan memasukkan semua objek *student* yang ada pada arraylist apakah method **addStudent** tetap berjalan.
- Pada **Then**, BDDMockito melakukan pengecekan objek baru apakah sesuai dengan objek check yang telah disiapkan.

Hasil testing :



3. Tambahkan *code* **deleteStudent** pada StudentServiceDatabase. Jelaskan apa yang terjadi pada method tersebut

```
@Test
public void deleteStudent () {
    StudentModel studentModel = new StudentModel("1506737823","Chanek",3.5);
    StudentModel check = new StudentModel ("1506737823","Chanek",3.5);
    BDDMockito.given(studentService.deleteStudent("1506737823")).willReturn(studentModel);

    //When
    StudentModel test = studentService.deleteStudent("1506737823");

    //Then
    BDDMockito.then(studentMapper).should().deleteStudent("1506737823");
    assertThat(test, equalTo(true)); //check if same
}
```

Jawab : pada method deleteStudent terdapat 3(tiga) segmen yaitu Given, When dan Then yang akan dilakukan pada setiap *testing*.

- Pada **Given**, pengguna menginisialisasi objek studentModels kemudian BDDMockito melakukan proses sesuai *method* **deleteStudent** dengan hasil *return value* sesuai dengan willReturn(studentModels).
- Pada **When**, akan dilakukan pengetesan terhadap method **deleteStudent** dengan memasukkan semua objek *student* yang ada pada arraylist apakah method **deleteStudent** tetap berjalan.
- Pada **Then**, BDDMockito melakukan pengecekan objek baru apakah sesuai dengan objek check yang telah disiapkan.

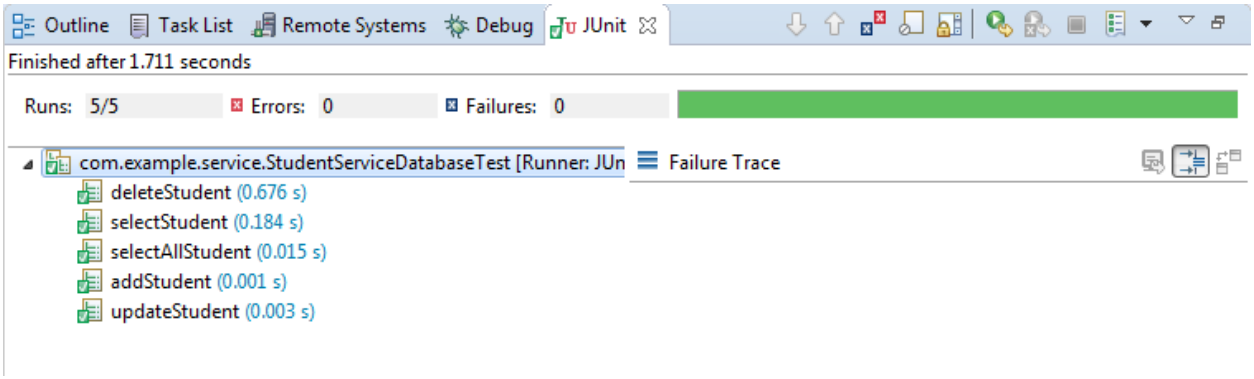
4. Tambahkan *code* **updateStudent** pada StudentServiceDatabase. Jelaskan apa yang terjadi pada method tersebut

```
@Test
public void updateStudent () {
    StudentModel studentModel = new StudentModel("1506737823","Chanek",3.5);
    StudentModel check = new StudentModel ("1506737823","Chanek",3.5);
    BDDMockito.given(studentService.updateStudent(studentModel)).willReturn(true);

    //When
    boolean test = studentService.updateStudent(studentModel);

    //Then
    BDDMockito.then(studentMapper).should().updateStudent(studentModel);
    assertThat(test, equalTo(true)); //check if same
}
```

Jawab : pada method updateStudent terdapat 3(tiga) segmen yaitu Given, When dan Then yang akan dilakukan pada setiap *testing*.

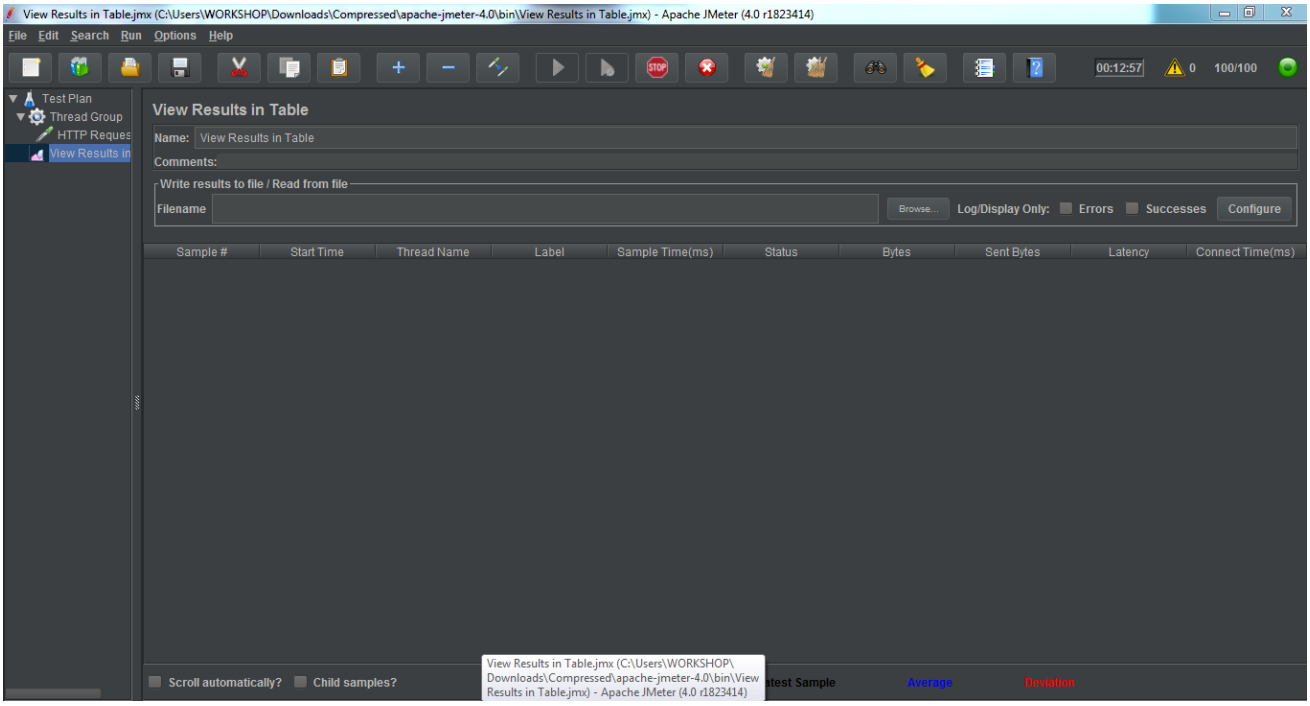
- Pada **Given**, pengguna menginisialisasi objek studentModels kemudian BDDMockito melakukan proses sesuai *method* **updateStudent** dengan hasil *return value* sesuai dengan willReturn(studentModels).
- Pada **When**, akan dilakukan pengetesan terhadap method **updateStudent** dengan memasukkan semua objek *student* yang ada pada arraylist apakah method **updateStudent** tetap berjalan.

- Pada **Then**, BDDMockito melakukan pengecekan objek baru apakah sesuai dengan objek check yang telah disiapkan.

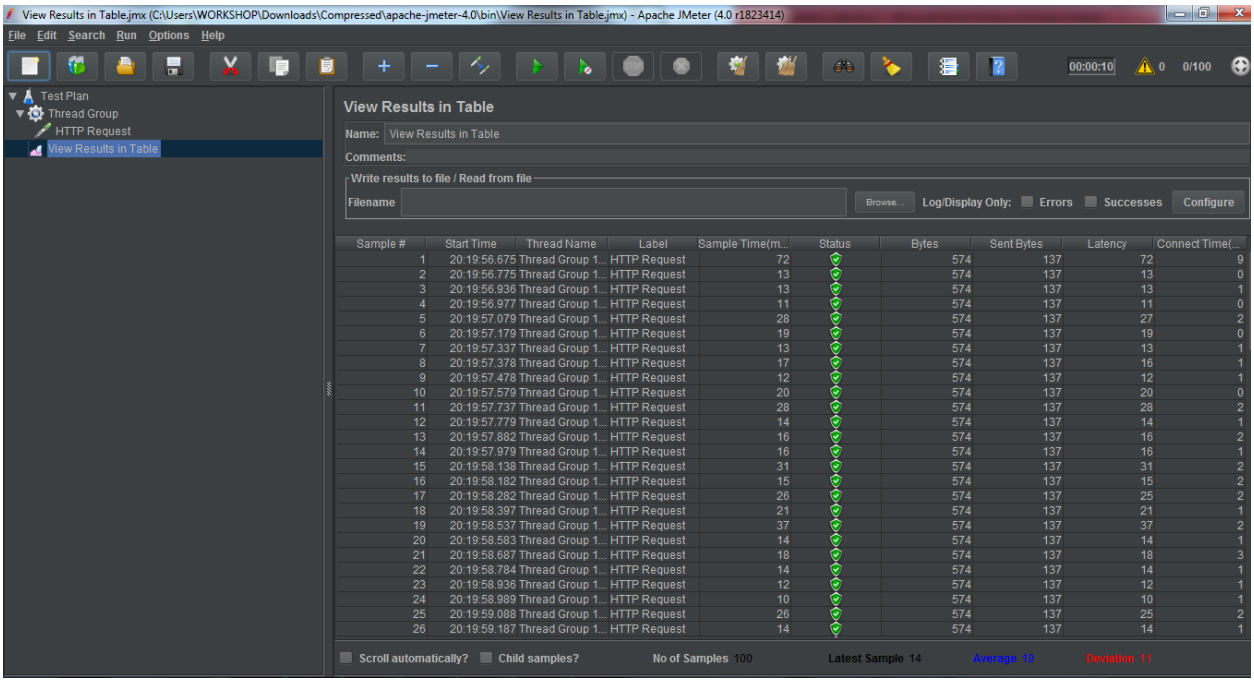


Load Testing Dengan Apache JMeter

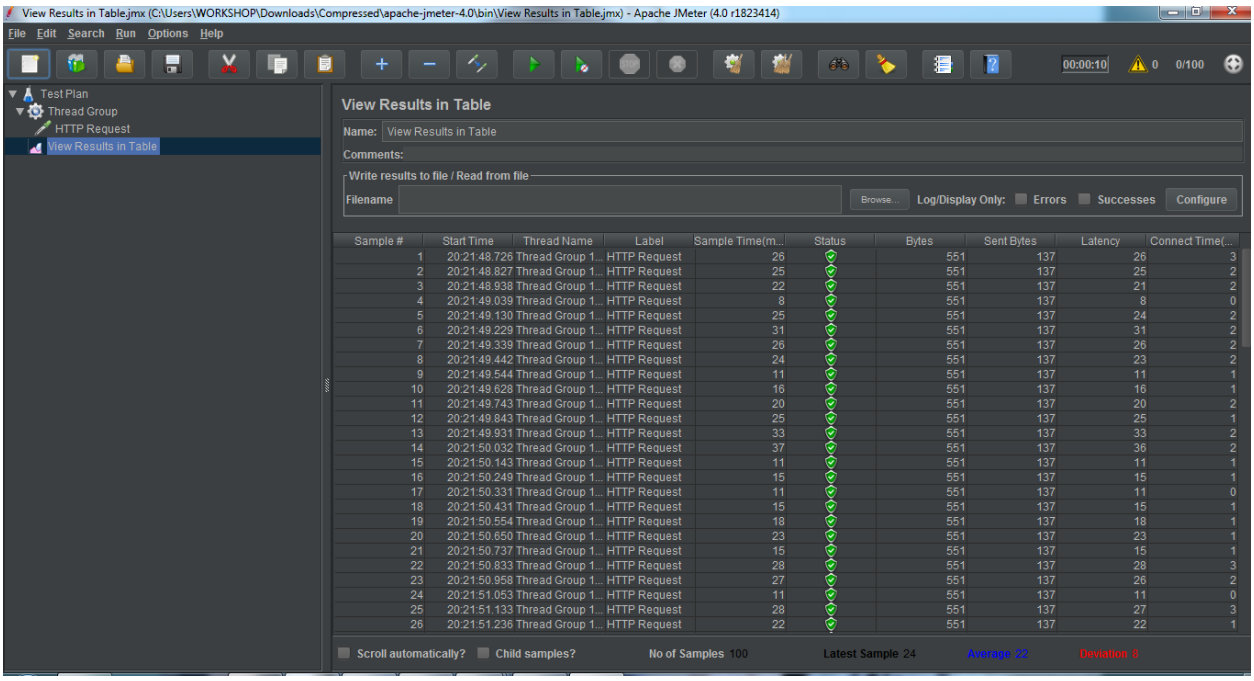
JMeter Select All Student berhasil ditemukan



JMeter Select student berhasil ditemukan

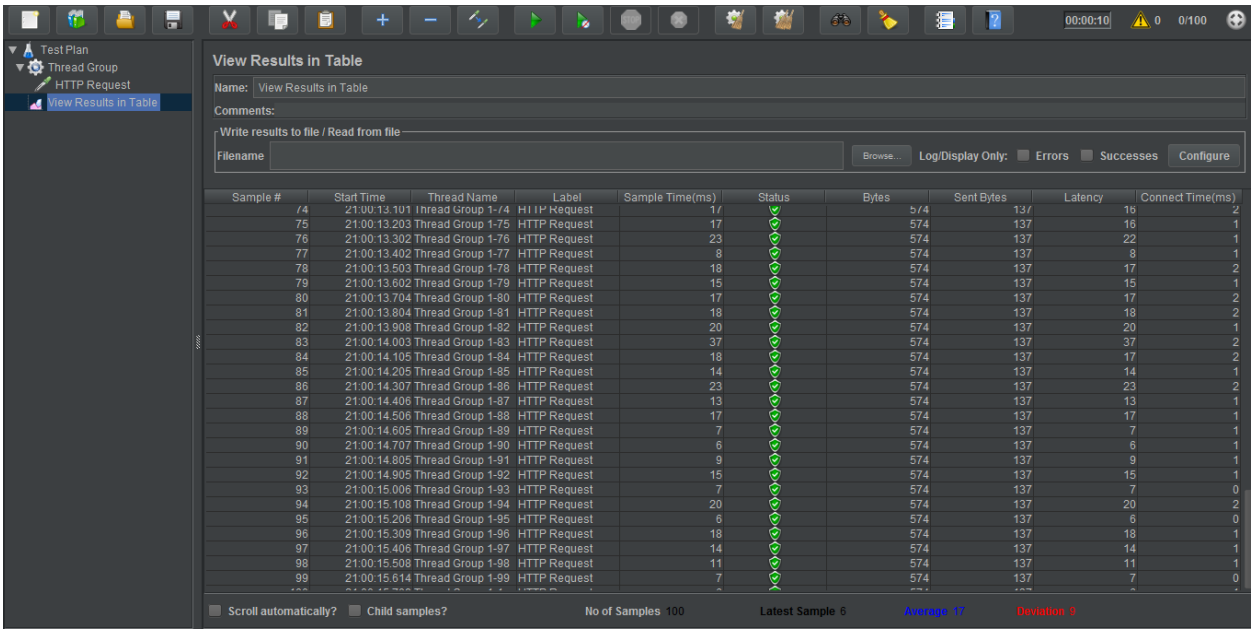


JMeter Select student dengan kondisi gagal ditemukan



Optimisasi dapat dilakukan pada basis data dengan cara membuat kolom npm menjadi unik dan indeks.

JMeter Select student berhasil ditemukan



JMeter Select student tidak berhasil ditemukan