



Back End Development 1

sesi 12



TESTING API SWAGGER + POSTMAN

How To Testing API Swagger + Postman

Pada Sesi kali ini kita akan berkenalan dengan proses testing, dan secara tidak langsung kita telah melakukan pengujian pada 3 sesi sebelumnya terkait CRUD Web API yang kita telah buat serta kita lakukan test terhadap setiap endpoint yang sudah kita buat baik itu di Postman ataupun Swagger.

Nah Jadi Proses Pengujian kali ini kita akan mengenal apa itu Testing , Jenis Testing, dan bagaimana Melakukan Testing untuk scope WEB API ini dengan beberapa tools support baik itu yang sudah dipersiapkan oleh Microsoft maupun external tools lainnya.



TESTING API SWAGGER + POSTMAN - Sesi 12

Testing Introduction

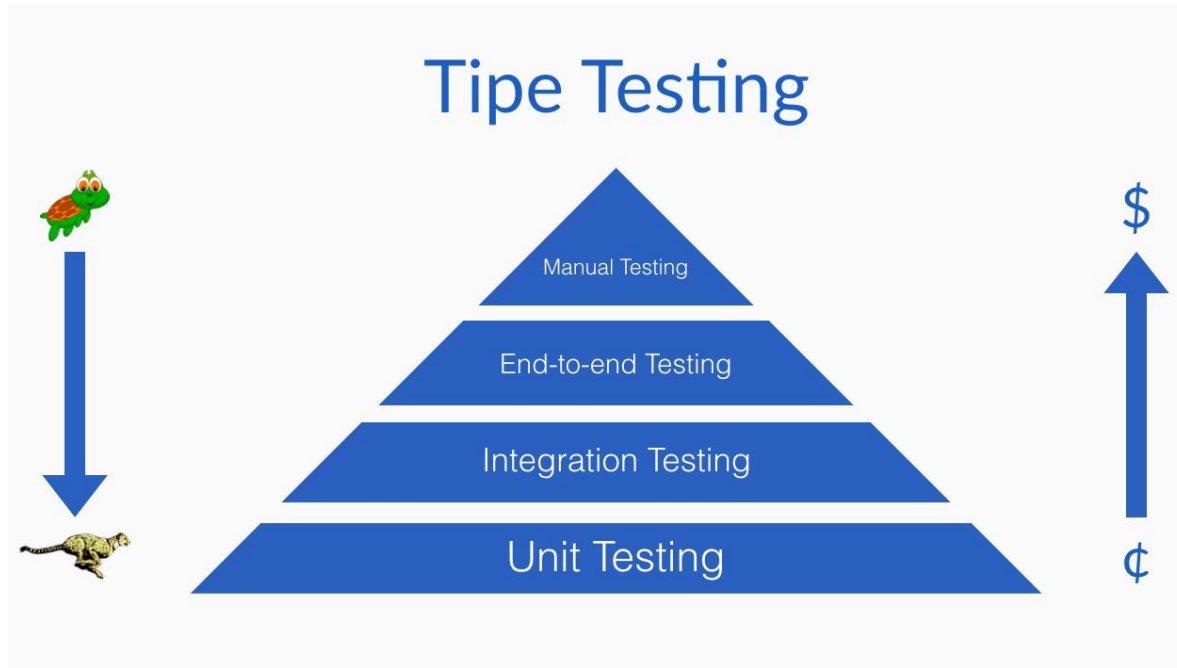
Testing kode yang sudah kita buat merupakan hal yang bisa dibilang wajib.

Tujuannya agar kode yang kita hasilkan selalu berkualitas tinggi dan merupakan sebuah garansi bahwa apa yang sudah kita kerjakan sesuai harapan, dan tentunya jumlah bugs atau kesalahan yang sangat minimal.

Saat ini, developer selain ngoding juga dituntut untuk juga bisa melakukan testing terhadap codingan kita sendiri.

Tipe Testing

Ada beberapa tipe atau jenis testing yang bisa kita lakukan untuk melakukan testing terhadap aplikasi atau komponen kita.



- Unit: Testing terhadap fungsi tunggal atau single functionality. Misalnya ada sebuah fungsi yang tugasnya mengambil data dari server. Yang dites Hanya satu fungsi itu saja. Tampilan dan lain sebagainya tidak dites dalam hal ini.
- Integration: Beberapa fungsi terkait dites secara bersamaan. Misalnya ada beberapa fungsi yang tugasnya mengambil data dari server dan menampilkan ke layar. Yang dites ya mulai dari ambil data hingga tampilan.
- End to end: Memastikan seluruh fungsi bekerja dari perspektif user. Jadi ngetest-nya dengan button click, apa yang terjadi dan seterusnya.

Dan seperti yang terlihat di ilustrasi diatas, unit testing merupakan jenis test yang paling cepat dijalankan. Dan semakin keatas proses eksekusi akan semakin lambat. Unit test juga merupakan solusi testing yang paling murah dibandingkan dengan solusi testing lain.

Untuk part Manual Testing ini sama seperti cara kita menguji langsung endpoint yang telah kita buat pada 3 sesi sebelumnya.



Unit Testing⁺ Xunit

Unit Testing Introduction

Membuat unit test bisa susah dalam hal waktu dan terkesan “lambat” jika kita tidak bisa memisahkan class yang ingin kita test dari keseluruhan system.

Pada sesi ini kita akan belajar membuat Mocking di .NET Core Unit Tests dengan Moq.

Kita akan mempelajari bagaimana membuat mocks dan menggunakannya sebagai dependency class yang ingin di test.

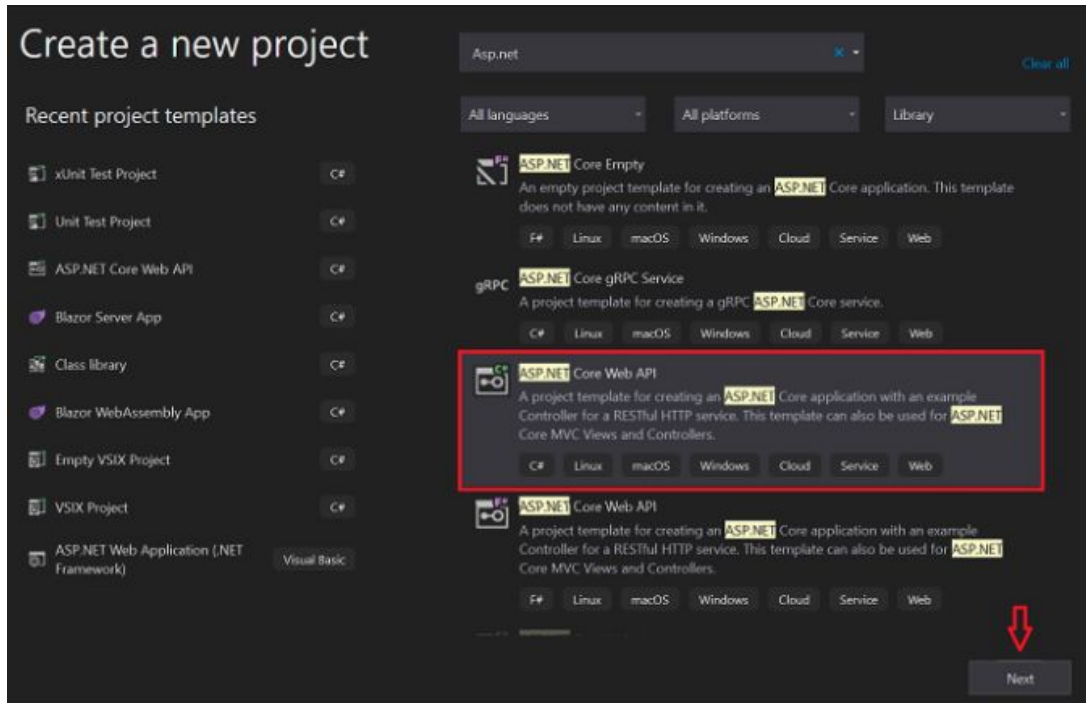
Pertama-tama kita harus membuat method dan property tiruan (mocking) untuk mengembalikan nilai tertentu, lalu bagaimana melakukan pengujian untuk menghasilkan sebuah interaksi dan terakhir kita akan explore bagaimana set-up exceptions & events dari proses mock tersebut.



TESTING API SWAGGER + POSTMAN - Sesi 12

Set up Projects

Buat Sample Web API Project dengan basic CRUD menggunakan pendekatan Ef Core.



Configure your new project

ASP.NET Core Web API

C#

Linux

macOS

Windows

Cloud

Service

Web

Project name

UnitTest_Mock



Define the name of Project

Location

C:\Users\mjayakr\source\repos

Solution

Create new solution

Solution name ⓘ

UnitTest_Mock



Place solution and project in the same directory

Back

Next




HACKTIV8

Kita akan menggunakan Template bawaan dari .NET 5.0

Additional information

ASP.NET Core Web API C# Linux macOS Windows Cloud Service Web

Target Framework

.NET 5.0 (Current) 

Authentication Type

None


☒ Configure for HTTPS

☐ Enable Docker

Docker OS

Linux

☒ Enable OpenAPI support



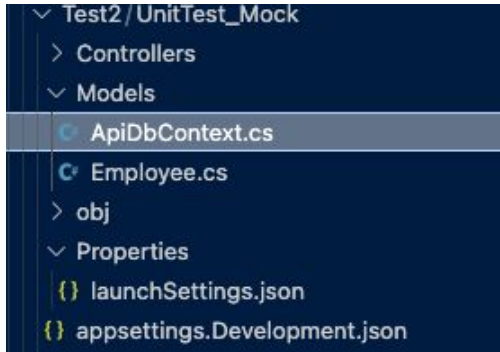
Back Create

Buat Folder Model dan didalamnya kita akan konfigurasi class Model dan DbContext seperti sesi sebelumnya menggunakan pendekatan Core Framework Entity



```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel.DataAnnotations;
4 using System.Linq;
5 using System.Threading.Tasks;
6
7 namespace UnitTest_Mock.Model
8 {
9     public class Employee
10     {
11         [Key]
12         public int Id { get; set; }
13         public string Name { get; set; }
14         public string Desgination { get; set; }
15     }
16 }
```

Masih dalam Folder Models buat 1 class bernama ApiDbContext.cs



```
1 {
2   "ConnectionStrings": {
3     "myconn": "server=localhost:1443; database=UnitTest;Trusted_Connection=True;"
4   },
5   "Logging": {
6     "LogLevel": {
7       "Default": "Information",
8       "Microsoft": "Warning",
9       "Microsoft.Hosting.Lifetime": "Information"
10    }
11  },
12  "AllowedHosts": "*"
13 }
14 }
```



TESTING API SWAGGER + POSTMAN - Sesi 12

appsettings.json

Selanjutnya buat string connection pada appsettings.json

```
1  {
2    "ConnectionStrings": {
3      "myconn": "server=localhost:1443; database=UnitTest;Trusted_Connection=True;"
4    },
5    "Logging": {
6      "LogLevel": {
7        "Default": "Information",
8        "Microsoft": "Warning",
9        "Microsoft.Hosting.Lifetime": "Information"
10     }
11   },
12   "AllowedHosts": "*"
13 }
14
```



TESTING API SWAGGER + POSTMAN - Sesi 12

startup.cs

```
1 using Microsoft.AspNetCore.Builder;
2 using Microsoft.AspNetCore.Hosting;
3 using Microsoft.AspNetCore.HttpsPolicy;
4 using Microsoft.AspNetCore.Mvc;
5 using Microsoft.EntityFrameworkCore;
6 using Microsoft.Extensions.Configuration;
7 using Microsoft.Extensions.DependencyInjection;
8 using Microsoft.Extensions.Hosting;
9 using Microsoft.Extensions.Logging;
10 using Microsoft.OpenApi.Models;
11 using System;
12 using System.Collections.Generic;
13 using System.Linq;
14 using System.Threading.Tasks;
15 using UnitTest_Mock.Model;
16 using UnitTest_Mock.Services;
17
18 namespace UnitTest_Mock
19 {
20     public class Startup
21     {
22         public Startup(IConfiguration configuration)
23         {
24             Configuration = configuration;
25         }
26
27         public IConfiguration Configuration { get; }
28
29         // This method gets called by the runtime. Use this method to add services to the container.
30         public void ConfigureServices(IServiceCollection services)
31         {
32             services.AddControllers();
33             services.AddSwaggerGen(c =>
34             {
35                 c.SwaggerDoc("v1", new OpenApiInfo { Title = "UnitTest_Mock", Version = "v1" });
36             });
37             #region Connection String
38             services.AddDbContext<AppDbContext>(item => item.UseSqlServer(Configuration.GetConnectionString("myconn")));
39             #endregion
40             services.AddScoped<IEmployeeService, EmployeeService>();
41         }
42     }
43 }
```



```
44 // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
45 public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
46 {
47     if (env.IsDevelopment())
48     {
49         app.UseDeveloperExceptionPage();
50         app.UseSwagger();
51         app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "UnitTest_Mock v1"));
52     }
53
54     app.UseHttpsRedirection();
55
56     app.UseRouting();
57
58     app.UseAuthorization();
59
60     app.UseEndpoints(endpoints =>
61     {
62         endpoints.MapControllers();
63     });
64 }
65 }
66 }
```



TESTING API SWAGGER + POSTMAN - Sesi 12

Migrasi Database

```
Maliks-MacBook-Air:UnitTest_Mock swijaya$ dotnet ef migrations add "initial"
```

```
Maliks-MacBook-Air:UnitTest_Mock swijaya$ dotnet ef database update
```



TESTING API SWAGGER + POSTMAN - Sesi 12

Migrasi Database

Buat Folder Services dimana kita bisa eksekusi business logic untuk semua operations.



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Threading.Tasks;
5 using UnitTest_Mock.Model;
6 using Microsoft.EntityFrameworkCore;
7
8 namespace UnitTest_Mock.Services
9 {
10     public class EmployeeService : IEmployeeService
11     {
12         #region Property
13         private readonly AppDbContext _appDbContext;
14         #endregion
15
16         #region Constructor
17         public EmployeeService(AppDbContext appDbContext)
18         {
19             _appDbContext = appDbContext;
20         }
21         #endregion
22
23         public async Task<string> GetEmployeebyId(int EmpID)
24         {
25             var name = await _appDbContext.Employees.Where(c=>c.Id == EmpID).Select(d=> d.Name).FirstOrDefaultAsync();
26             return name;
27         }
28
29         public async Task<Employee> GetEmployeeDetails(int EmpID)
30         {
31             var emp = await _appDbContext.Employees.FirstOrDefaultAsync(c => c.Id == EmpID);
32             return emp;
33         }
34     }
35 }
```



TESTING API SWAGGER + POSTMAN - Sesi 12

IEmployeeService.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using UnitTest_Mock.Model;
6
7  namespace UnitTest_Mock.Services
8  {
9      public interface IEmployeeService
10     {
11         Task<string> GetEmployeebyId(int EmpID);
12         Task<Employee> GetEmployeeDetails(int EmpID);
13     }
14 }
```



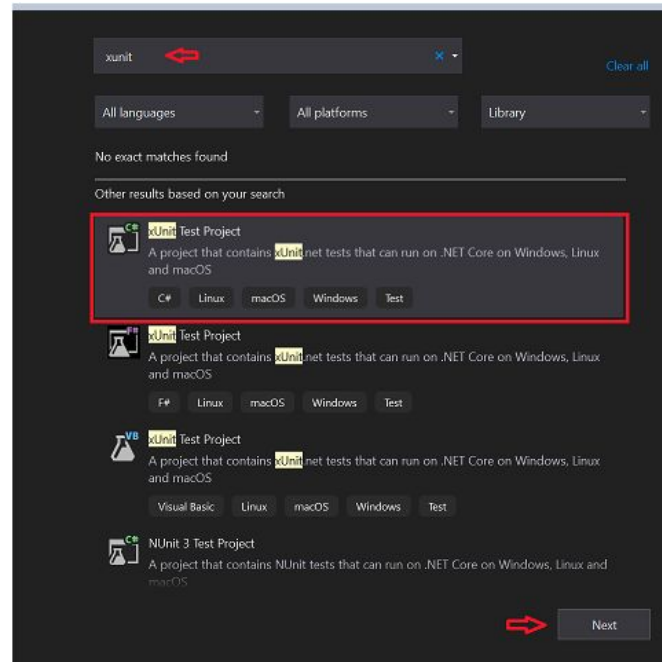
Buat Method API untuk service pada class controller

```
1 using Microsoft.AspNetCore.Mvc;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Threading.Tasks;
6 using UnitTest_Mock.Model;
7 using UnitTest_Mock.Services;
8
9 namespace UnitTest_Mock.Controllers
10 {
11     [Route("api/[controller]")]
12     [ApiController]
13     public class EmployeeController : ControllerBase
14     {
15         #region Property
16         private readonly IEmployeeService _employeeService;
17         #endregion
18
19         #region Constructor
20         public EmployeeController(IEmployeeService employeeService)
21         {
22             _employeeService = employeeService;
23         }
24         #endregion
25
26         [HttpGet(nameof(GetEmployeeById))]
27         public async Task<string> GetEmployeeById(int EmpID)
28         {
29             var result = await _employeeService.GetEmployeebyId(EmpID);
30             return result;
31         }
32         [HttpGet(nameof(GetEmployeeDetails))]
33         public async Task<Employee> GetEmployeeDetails(int EmpID)
34         {
35             var result = await _employeeService.GetEmployeeDetails(EmpID);
36             return result;
37         }
38     }
39 }
40 }
```



Silahkan buat project testing baru didalam project solution dimana kita akan menulis test case untuk setiap fungsi

1. Klik Kanan pada Solution
2. Klik Add - New Project
3. Search X-Unit Test Projects



Configure your new project

xUnit Test Project

C#

Linux

macOS

Windows

test

Project name

TestProject1



Name the Project

Location

C:\Users\mjayakr\source\repos\UnitTest_Mock



Back

Next



HACKTIV8

Choose the target framework the same as where we have used it in our API project.

Additional information

xUnit Test Project C# Linux macOS Windows Test

Target Framework

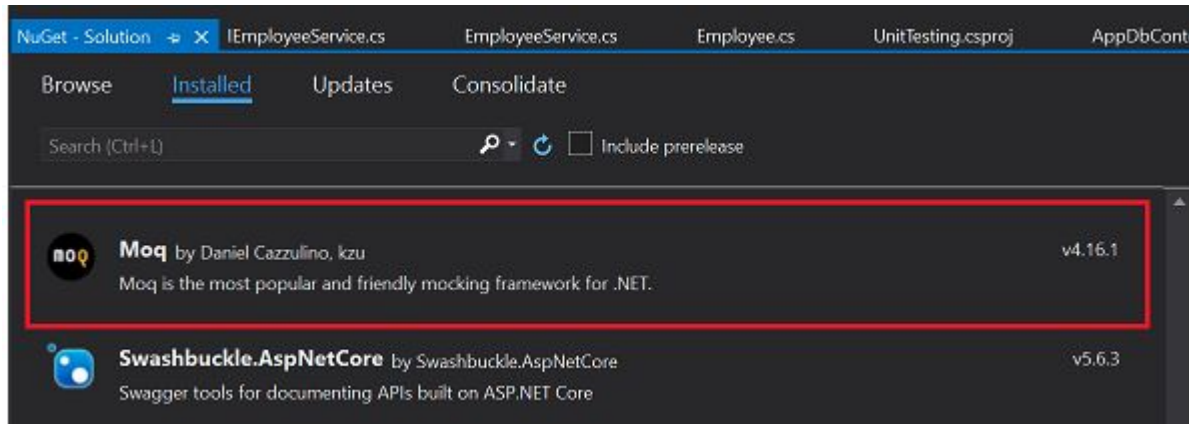
.NET 5.0 (Current) ←

Back Create



HACKTIV8

Install the **Moq** package inside this unit test project.



Buat Class di dalam proyek Test ini untuk mendefinisikan semua masing2 test case kita, Terlebih dahulu harus memasukkan data ke dalam tabel yang telah kita buat.

Buka SQL Server dan masukkan data dummy ke tabel karyawan.


```

1  using Moq;
2  using UnitTest_Mock.Controllers;
3  using UnitTest_Mock.Model;
4  using UnitTest_Mock.Services;
5  using Xunit;
6
7  namespace UnitTesting
8  {
9      public class EmployeeTest
10     {
11         #region Property
12         public Mock<IEmployeeService> mock = new Mock<IEmployeeService>();
13         #endregion
14
15         [Fact]
16         public async void GetEmployeebyId()
17         {
18             mock.Setup(p => p.GetEmployeebyId(1)).ReturnsAsync("JK");
19             EmployeeController emp = new EmployeeController(mock.Object);
20             string result = await emp.GetEmployeeById(1);
21             Assert.Equal("JK", result);
22         }
23
24         [Fact]
25         public async void GetEmployeeDetails()
26         {
27             var employeeDTO = new Employee()
28             {
29                 Id = 1,
30                 Name = "JK",
31                 Desgination = "SDE"
32             };
33             mock.Setup(p => p.GetEmployeeDetails(1)).ReturnsAsync(employeeDTO);
34             EmployeeController emp = new EmployeeController(mock.Object);
35             var result = await emp.GetEmployeeDetails(1);
36             Assert.True(employeeDTO.Equals(result));
37         }
38     }

```



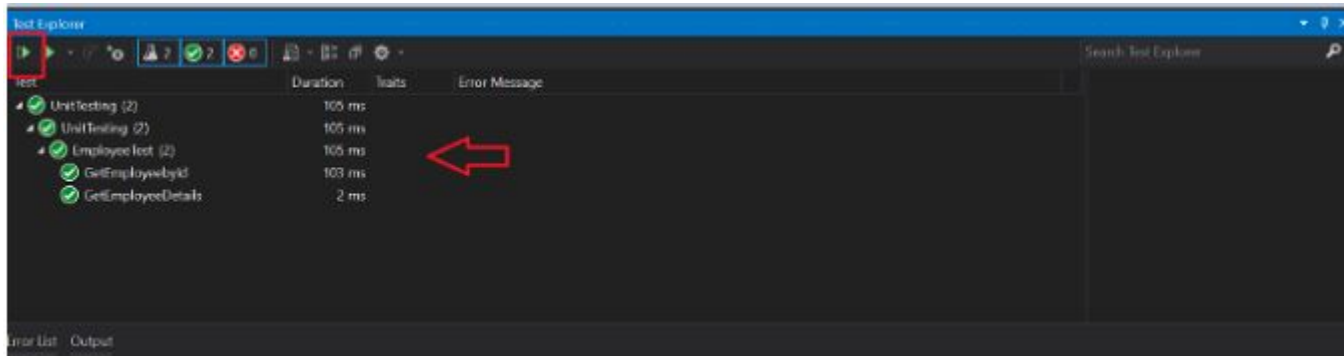
Kita siapkan mock untuk service API Bisnis kita pada level controller untuk cek hasil dan membandingkan dengan nilai yang dimasukkan oleh user.

Kita juga dapat nge-debug test cases untuk cek output di running mode.


Dengan cara : Larikan semua test cases untuk
we can debug the test cases to check the output in running mode.

Jalankan semua kasus test untuk memverifikasi apakah lolos atau gagal.

- Click on View in the Top left
- Click on Test explorer.



Pada gambar di atas, kita dapat melihat semua test case kita lulus termasuk durasi waktunya juga.



Performance Web Testing

TESTING API SWAGGER + POSTMAN - Sesi 12

Unit Test

Pada dasarnya unit test merupakan bagian dari white box testing, karena eksekusi testing dilakukan oleh programmer.

Mengingat tujuan awal tech-stack dengan menggunakan API nantinya akan digunakan oleh banyak client dan user, tentu alangkah baiknya kita menggunakan Performance Test Tool untuk proses ini.

Nah pada praktik kali ini kita akan menggunakan JMeter.

The Apache JMeter™ application is open source software, a 100% pure Java application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions

Sources : <http://jmeter.apache.org/>

TESTING API SWAGGER + POSTMAN - Sesi 12

Download JMeter dan extract ke drive

http://jmeter.apache.org/download_jmeter.cgi

Apache JMeter 5.4.1 (Requires Java 8+)

Binaries

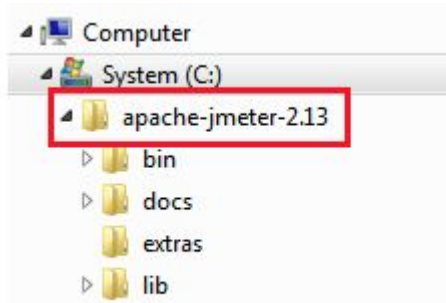
[apache-jmeter-5.4.1.tgz sha512 pgp](#)

[apache-jmeter-5.4.1.zip sha512 pgp](#)

Source

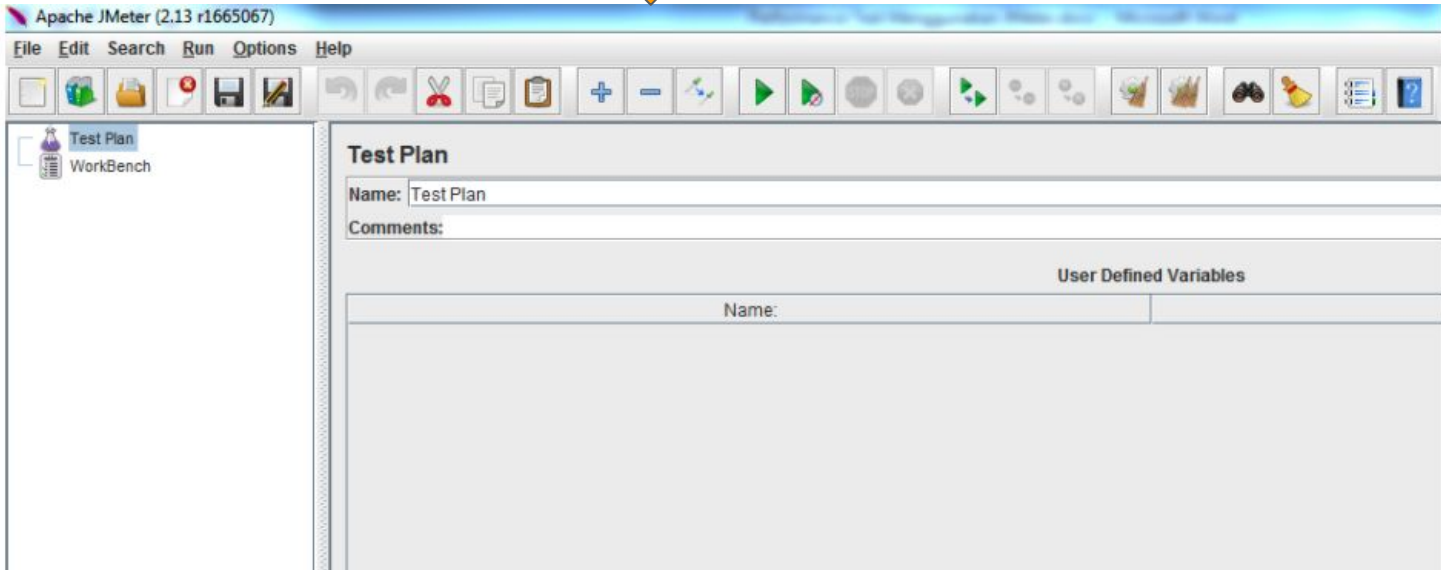
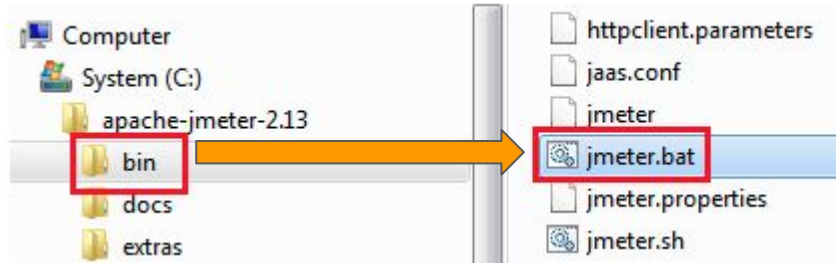
[apache-jmeter-5.4.1_src.tgz sha512 pgp](#)

[apache-jmeter-5.4.1_src.zip sha512 pgp](#)



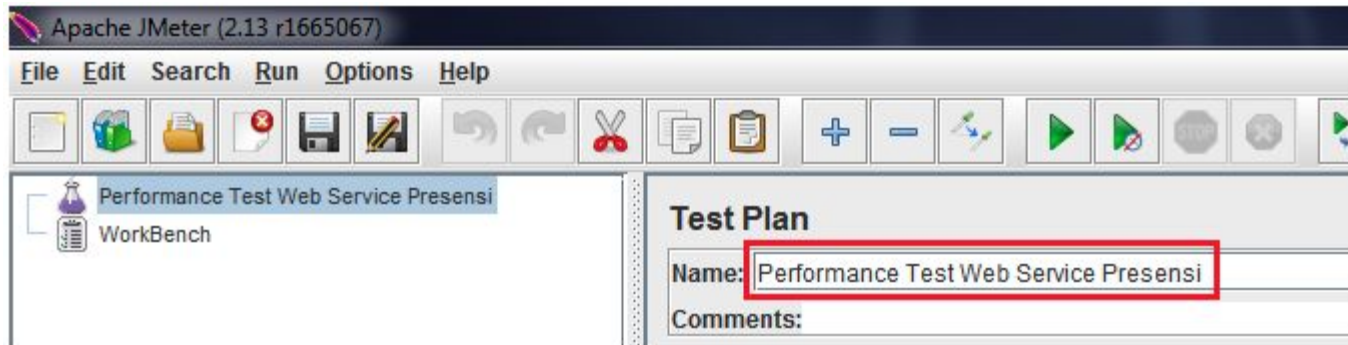
Extract dalam drive C:\

2. Untuk Jalankan JMeter, klik 2x file JMeter.bat yang terdapat pada folder bin



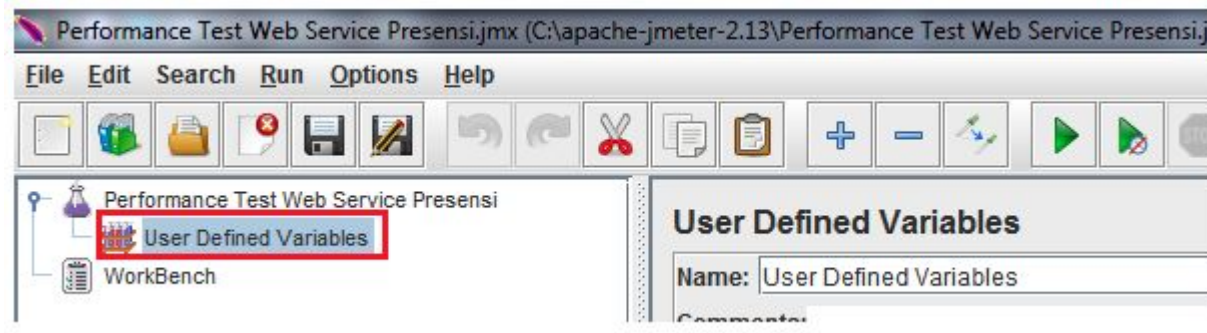
Coba ganti informasi Name dan diharapkan disesuaikan dengan kebutuhan,

Contoh Performance Test Web Service Presensi



TESTING API SWAGGER + POSTMAN - Sesi 12

User Defined Variables

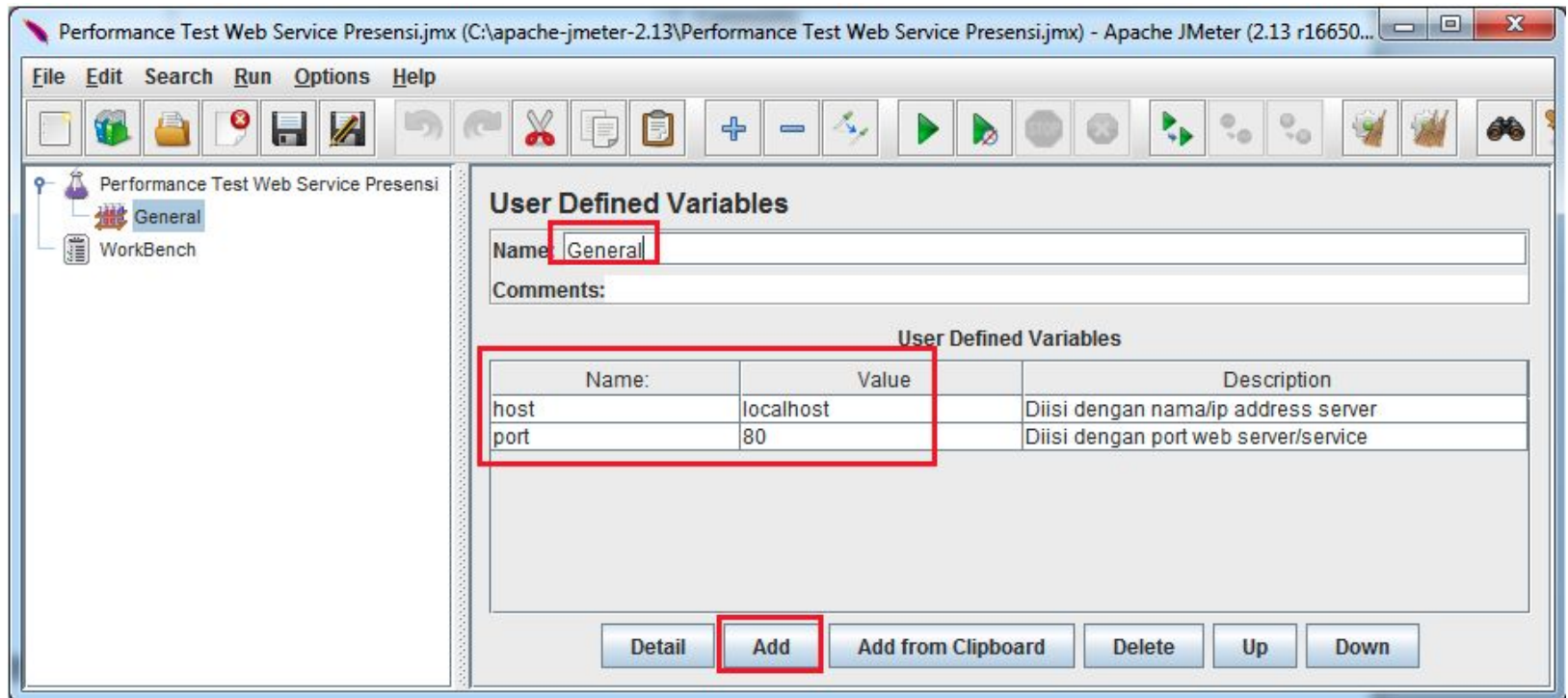


Di node ini kita akan menambahkan informasi global yang sering digunakan pada saat testing seperti informasi host dan port.

Untuk menambahkan node User Defined Variables :

klik kanan node Test Plan (Performance Test Web Service Presensi) -> Add -> Config Element -> User Defined Variables.

Kemudian lakukan pengaturan User Defined Variables seperti gambar berikut :



Pada gambar di atas kita menambahkan dua variabel yaitu host dan port.
Hint : Variabel-variabel ini nantinya akan digunakan pada langkah-langkah berikutnya.

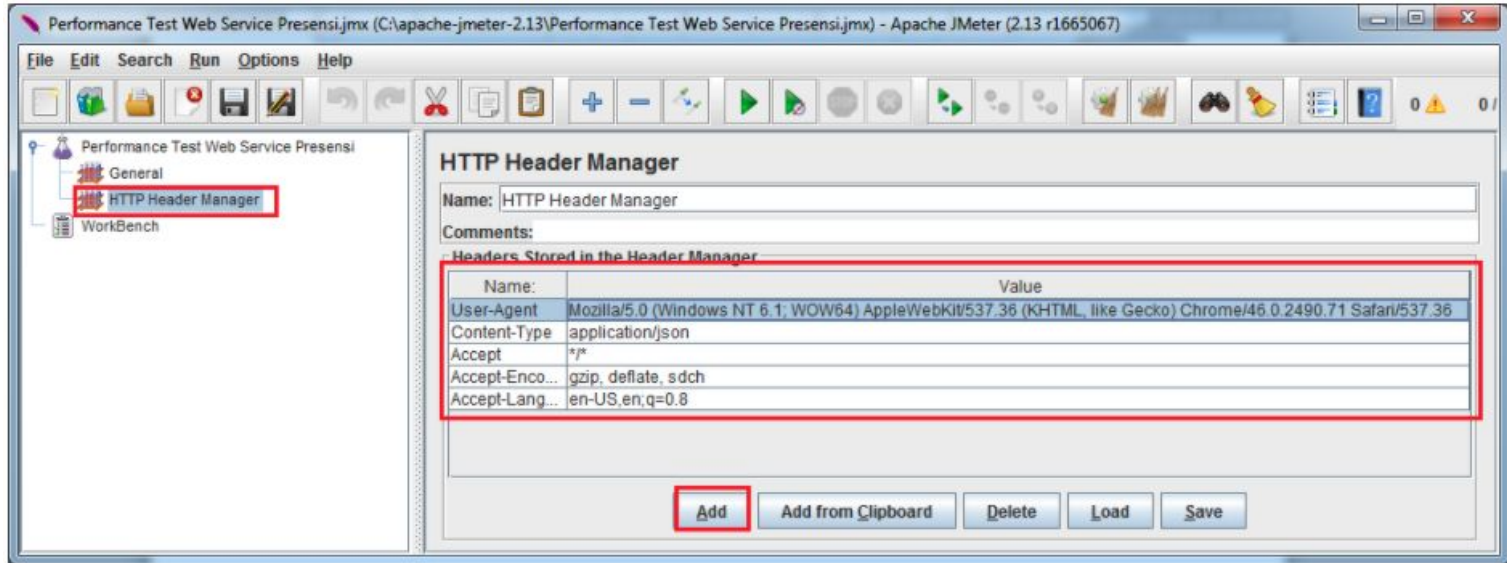
TESTING API SWAGGER + POSTMAN - Sesi 12

HTTP Header Manager

Kita lanjutkan dengan menambahkan node HTTP Header Manager.

Langkah-langkahnya sama seperti menambahkan node `User Defined Variables`, hanya saja yang dipilih adalah HTTP Header Manager`.

Di node ini kita akan menambahkan informasi apa saja yang dikirimkan [JMeter](#) ke HTTP request header.

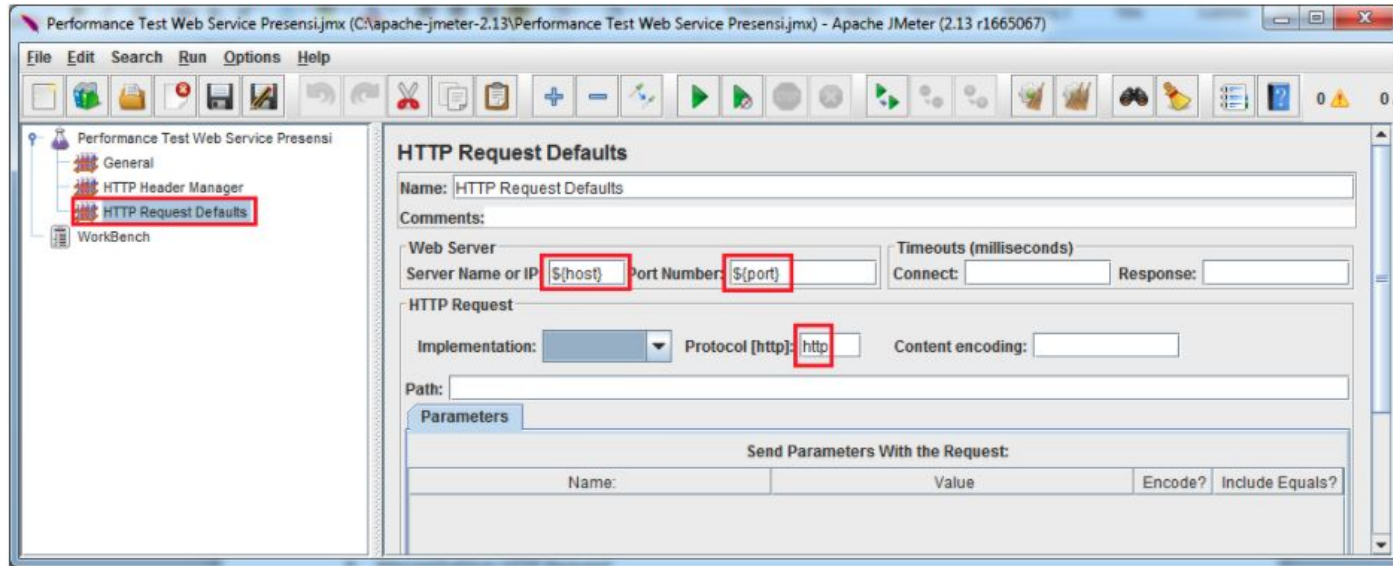


TESTING API SWAGGER + POSTMAN - Sesi 12

HTTP Request Defaults

Langkah berikutnya adalah menambahkan node HTTP Request Defaults, caranya juga sama seperti sebelumnya hanya saja yang dipilih node HTTP Request Defaults.

Di node ini kita cukup mengeset informasi nama server/ip address, port dan protocol.



Karena sebelumnya kita sudah mendefinisikan nama host dan port di node User Defined Variables, di node ini kita tinggal panggil variabel tersebut dengan format : `${NAMA_VARIABEL}`.

TESTING API SWAGGER + POSTMAN - Sesi 12

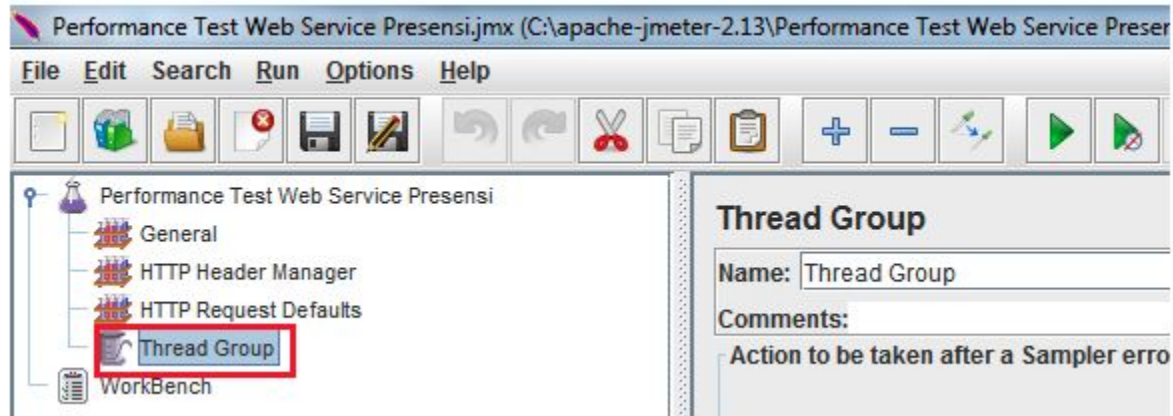
Thread Group

Node ini digunakan untuk mengelompokkan service yang akan di tes.

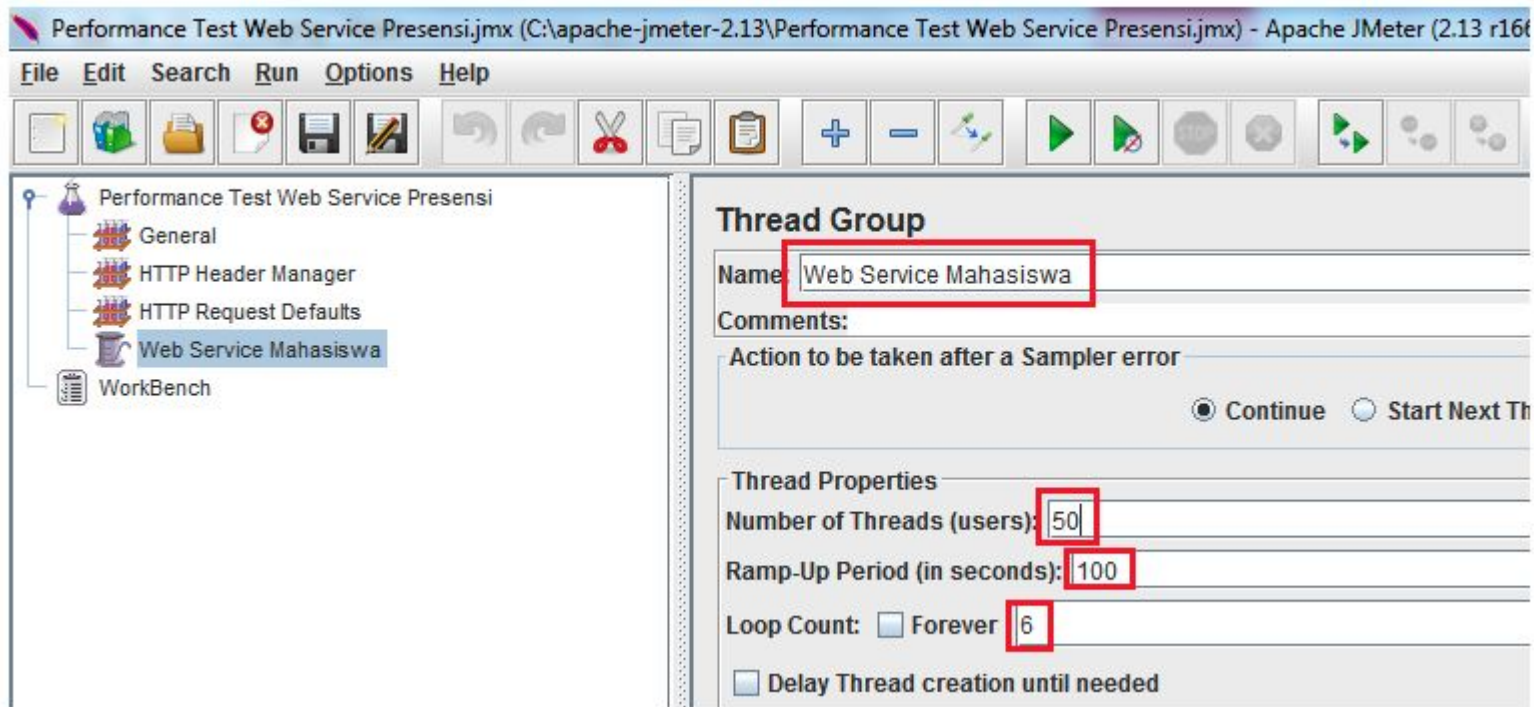
Misal kita mempunyai service Mahasiswa dan Dosen, kedua service ini sebaiknya dibuatkan Thread Group masing-masing.

Untuk menambahkan node Thread Group :

Klik kanan node Test Plan (Performance Test Web Service Presensi) -> Add -> Threads (Users) -> Thread Group.



Kemudian atur settingnya seperti berikut :



Dari setting di atas, kita akan membuat skenario performance test seperti berikut :

- Jumlah user sebanyak 50 orang
- Setiap 2 detik (100/50), akan mengirimkan 6 request ke server.
- Total jumlah sample = 300 (50 x 6)

Skenario performance test ini bisa diganti-ganti nilainya sesuai kebutuhan.

7. HTTP Request

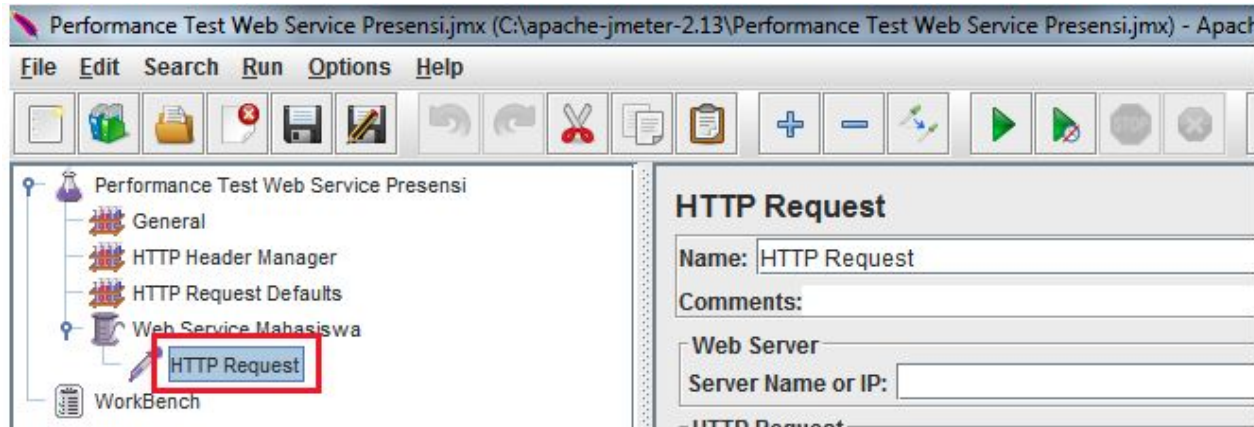
Setelah menentukan skenario performance test,
Langkah berikutnya adalah menambahkan node HTTP Request.

Di node inilah kita akan menentukan web service yang akan di tes. Misal web service mahasiswa mempunyai dua layanan yaitu GetByID dan GetByName.

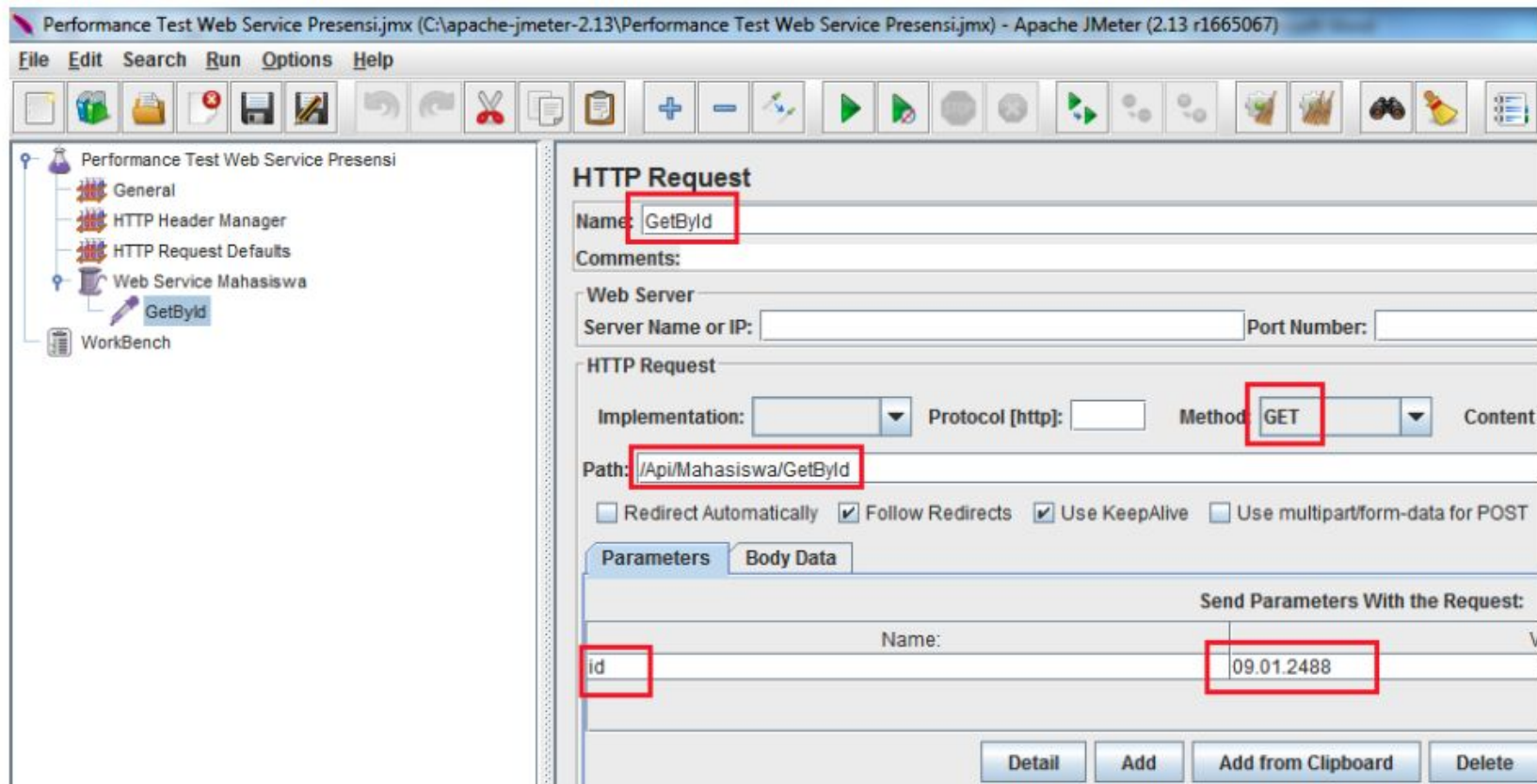
Nah dua service ini akan kita tambahkan sebagian dari HTTP Request.

Untuk menambahkan node HTTP Request :

Klik kanan node Thread Group (Web Service Mahasiswa) -> Add -> Sampler -> HTTP Request.



Kemudian atur settingnya seperti berikut :

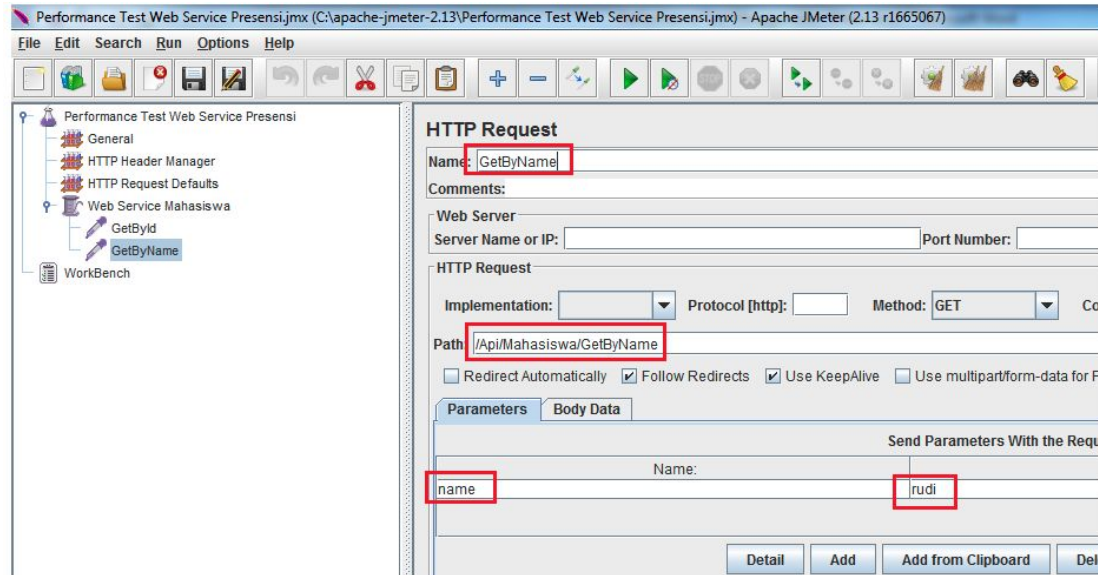


HACKTIV8

Dari gambar di atas, informasi web servicenya sebagai berikut :

- Nama service : GetById
- Path/Url : /Api/Mahasiswa/GetById
- Service ini mempunyai satu parameter yaitu id, yang kita set nilainya 09.01.2488
- Informasi server name/ip dan port number tidak perlu di set, karena sudah kita set nilainya di node HTTP Request Defaults

Ulangi langkah sebelumnya untuk menambahkan service yang lain.



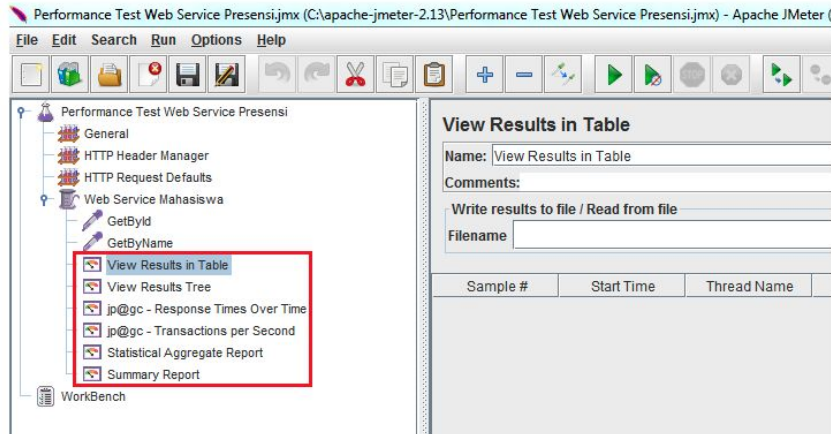
TESTING API SWAGGER + POSTMAN - Sesi 12

Performance Test Results

Ada beberapa format laporan yang digunakan untuk menampilkan hasil performance test yaitu :

- View Results in Table
- View Results Tree
- Response Times Over Time
- Transactions per Second
- Statistical Aggregate Report
- Summary Report

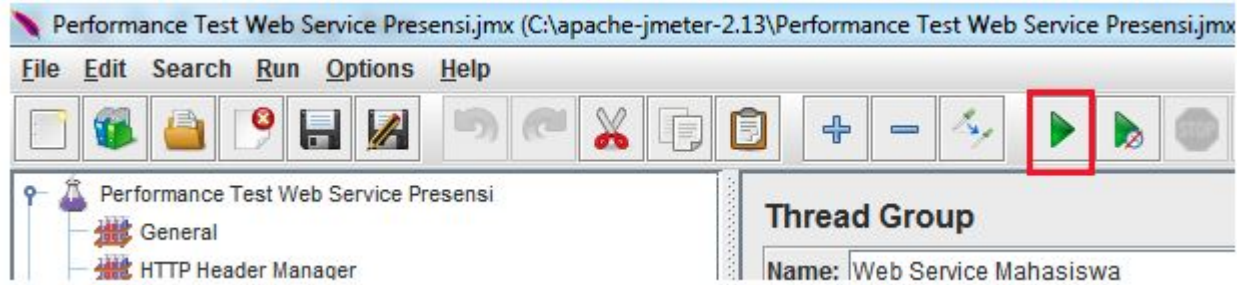
Untuk menambahkan semua format laporan di atas klik kanan node Thread Group (Web Service Mahasiswa) -> Add -> Listener -> Pilih jenis laporan.



TESTING API SWAGGER + POSTMAN - Sesi 12

Running Performance Test

Untuk menjalankan performance test, kita tinggal mengklik toolbar Start



Performance test ini di jalankan menggunakan Environment Test sebagai berikut :

Server Web Service :

- Sistem Operasi : Windows Server 2008 64 Bit
- Web Server : IIS 7
- Teknologi/framework REST Service : ASP.NET Web API 2
- Processor : Intel Xeon X5570 2.93 Ghz
- RAM : 12 GB

Server Database :

- Sistem Operasi : Windows Server 2012 64 Bit
- Database : SQL Server 2014
- Processor : Intel Xeon E5-2620 v2 2.10 Ghz
- RAM : 24 Gb



TESTING API SWAGGER + POSTMAN - Sesi 12

Results of Performance Test

Performance Test Web Service Presensi.jmx (C:\apache-jmeter-2.13\Performance Test Web Service Presensi.jmx) - Apache JMeter (2.13 r1665067)

File Edit Search Run Options Help

View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename Browse... Log/Display Only: ☐ Errors

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Latency
1	09:03:16.625	Web Service Mahasiswa 1-1	GetById	42		332	42
2	09:03:16.674	Web Service Mahasiswa 1-1	GetById	27		332	27
3	09:03:16.705	Web Service Mahasiswa 1-1	GetById	22		332	22
4	09:03:16.732	Web Service Mahasiswa 1-1	GetById	22		332	22
5	09:03:16.759	Web Service Mahasiswa 1-1	GetById	21		332	21
6	09:03:16.786	Web Service Mahasiswa 1-1	GetById	29		332	29
7	09:03:18.626	Web Service Mahasiswa 1-2	GetById	30		332	30
8	09:03:18.662	Web Service Mahasiswa 1-2	GetById	28		332	28
9	09:03:18.700	Web Service Mahasiswa 1-2	GetById	31		332	31



Performance Test Web Service Presensi.jmx (C:\apache-jmeter-2.13\Performance Test Web Service Presensi.jmx) - Apache JMeter (2.13 r1665067)

File Edit Search Run Options Help

Performance Test Web Service Presensi

- General
- HTTP Header Manager
- HTTP Request Defaults
- Web Service Mahasiswa
 - GetById
 - GetByName
 - View Results in Table
 - View Results Tree**
 - jp@gc - Response Times Over Time
 - jp@gc - Transactions per Second
 - Statistical Aggregate Report
 - Summary Report
- WorkBench

View Results Tree

Name: View Results Tree

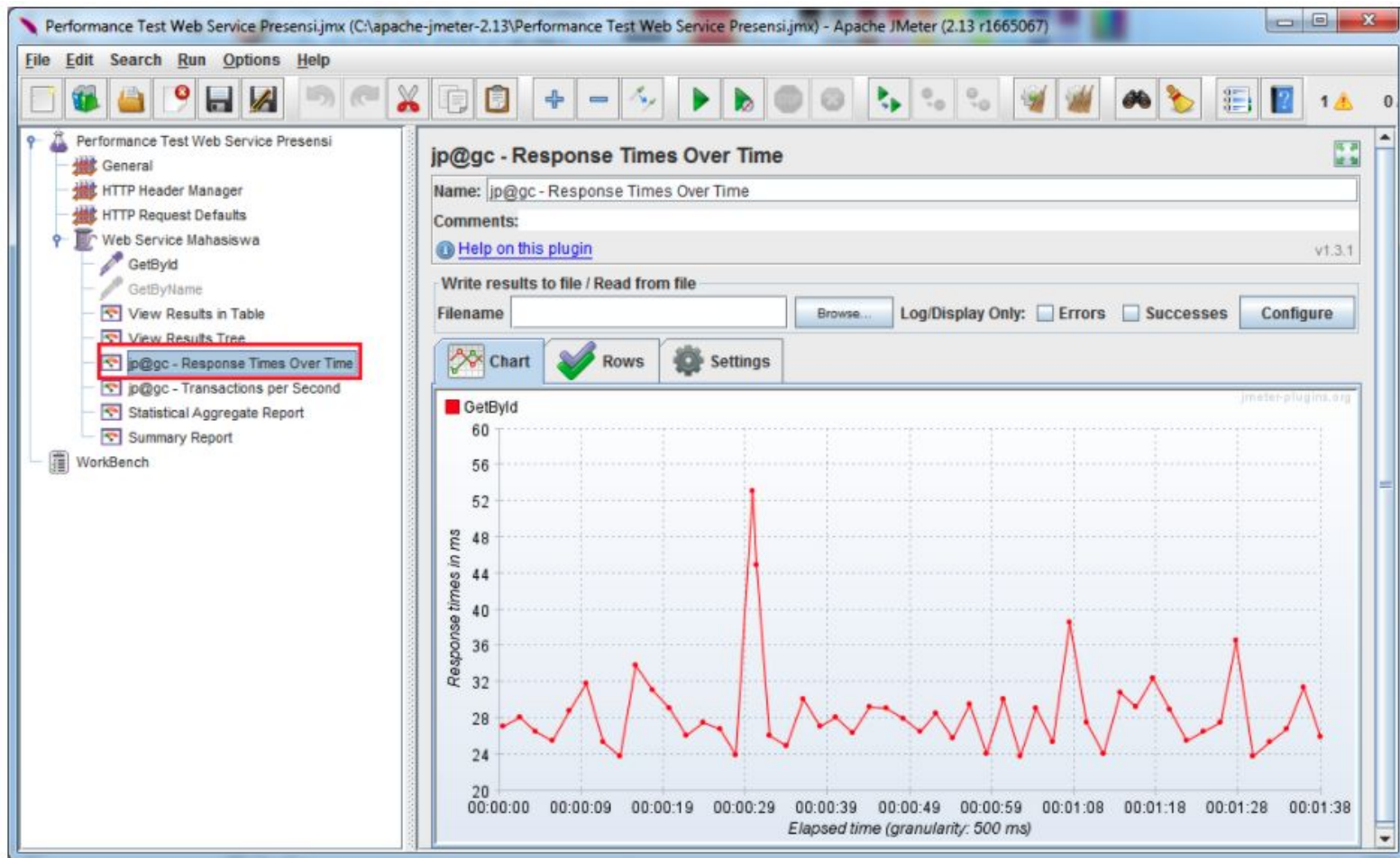
Comments:

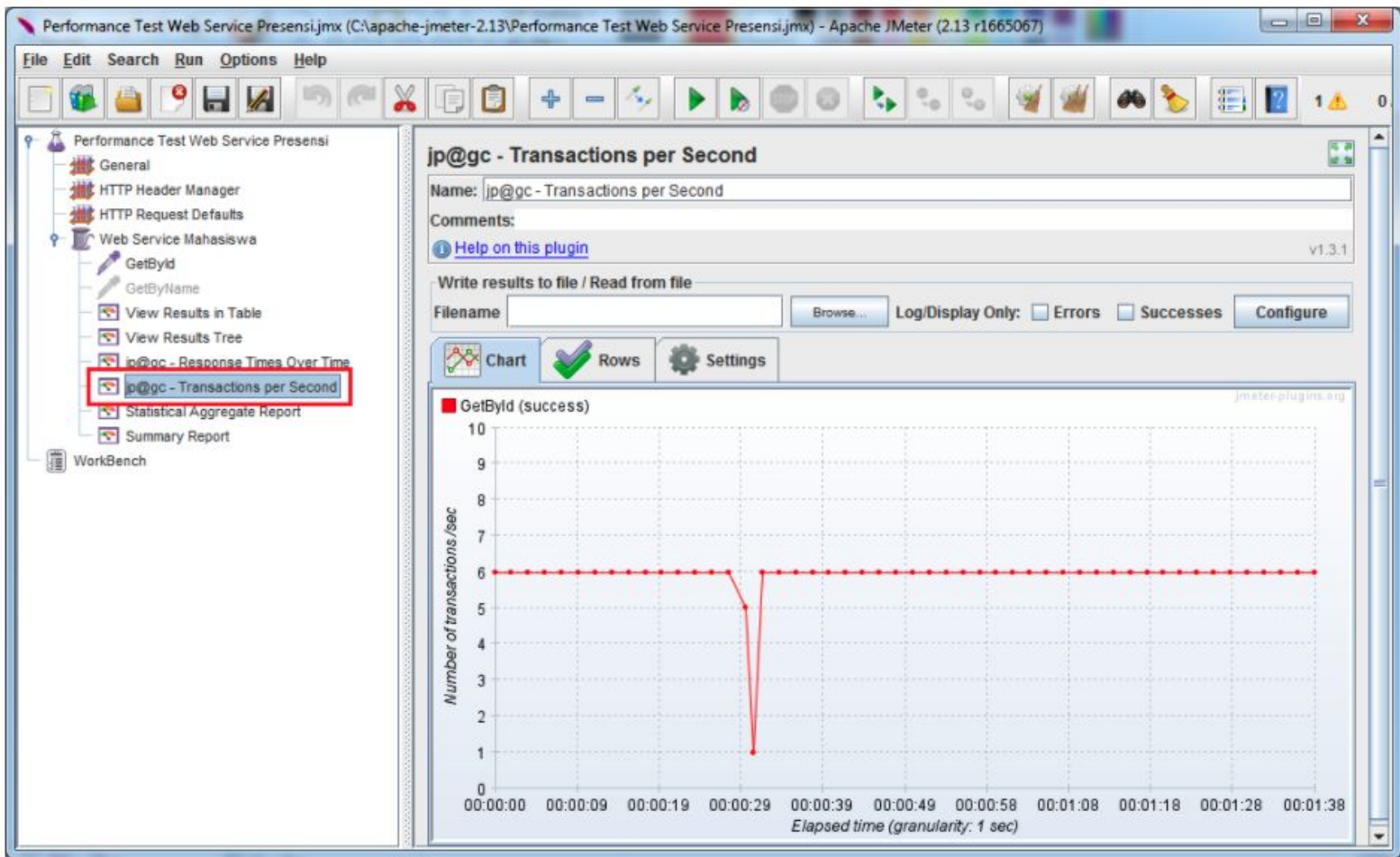
Write results to file / Read from file

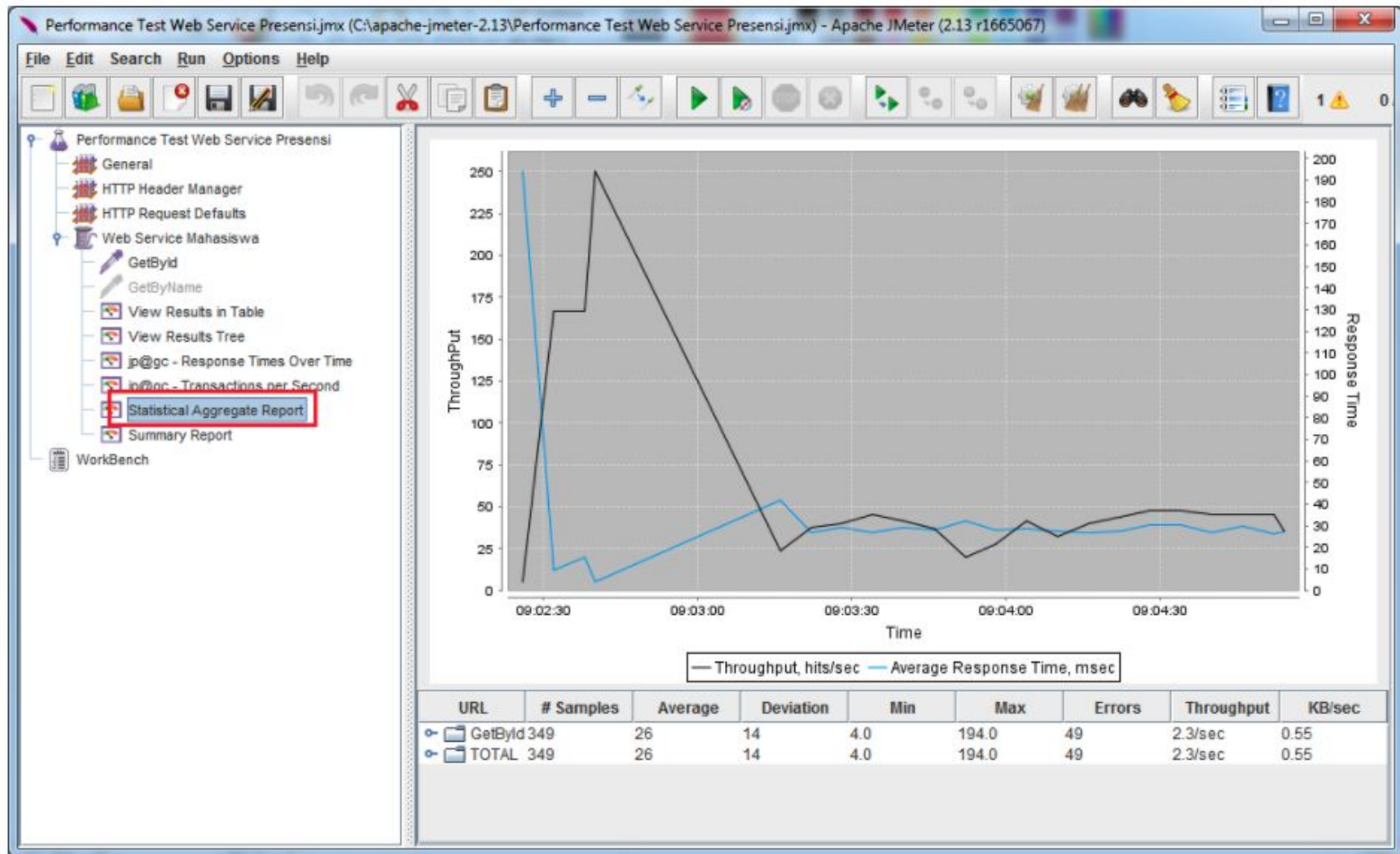
Filename

Text	Sampler result	Request	Response data
GetById	Thread Name: Web Service Mahasiswa 1-1		
GetById	Sample Start: 2015-10-20 09:03:16 ICT		
GetById	Load time: 22		
GetById	Connect Time: 0		
GetById	Latency: 22		
GetById	Size in bytes: 332		
GetById	Headers size in bytes: 258		
GetById	Body size in bytes: 74		
GetById	Sample Count: 1		
GetById	Error Count: 0		
GetById	Response code: 200		
GetById	Response message: OK		
GetById	Response headers:		
GetById	HTTP/1.1 200 OK		
GetById	Cache-Control: no-cache		
GetById	Pragma: no-cache		
GetById	Content-Type: application/json; charset=utf-8		









Performance Test Web Service Presensi.jmx (C:\apache-jmeter-2.13\Performance Test Web Service Presensi.jmx) - Apache JMeter (2.13 r1665067)

File Edit Search Run Options Help

Performance Test Web Service Presensi

- General
- HTTP Header Manager
- HTTP Request Defaults
- Web Service Mahasiswa
 - GetById
 - GetByName
 - View Results in Table
 - View Results Tree
 - jp@gc - Response Times Over Time
 - jp@gc - Transactions per Second
 - Statistical Aggregate Report
 - Summary Report**
- WorkBench

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename Browse... Log/Display Only: ☐ Errors ☐ Successes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
GetById	349	26	4	194	14.17	14.04%	2.3/sec	1.11	484.0
TOTAL	349	26	4	194	14.17	14.04%	2.3/sec	1.11	484.0



Untuk menemukan referensi yang tepat dalam menganalisa hasil performance test dapat dilihat pada :
Node View Results in Table Kolom Status.

Dari node ini kelihatan apakah ada request service yang berstatus WARNING

Jika ada, ini menjadi acuan kita untuk improve service atau bahkan mengubah skenario performance test dikarenakan faktor lain seperti kondisi jaringan internet yang tidak bagus pada saat melakukan test.