

Create a new Vert.x application

Version	<div>4.2.6 4.3.0-SNAPSHOT</div>
Language	<div>Java Kotlin</div>
Build	<div>Maven Gradle</div>
Group Id	<div>com.shree.calculator</div>
Artifact Id	<div>vertx-calculator</div>
Dependencies (2/78)	<div>Web, MQTT, etc.</div>
+ Show dependencies panel	
Advanced options —	
<hr/>	
Package	<div>Your project package name</div>
JDK Version	<div>JDK 1.8 JDK 11 JDK 17</div>
<hr/>	
Selected dependencies (2)	
<div>Vert.x Web × Vert.x Web Client ×</div>	
<div>Generate Project alt + ↵ >_</div>	

[→ Find a Vert.x how-to](#)

[🐞 Report an issue](#)

There was a problem with the JDK version too, and the error was showing me to downgrade the JDK version (After I chose Amazon corretto)

Then I saw all the vertx files are based on JDK 13 (even though I selected JDK 11 in vertx.io), so I selected the JDK 13 and re ran the project and it is working fine!

Creating a vertx calculator application

1. Methods to implement

Add | /add/:num1/:num2

Sub | /sub/:num1/:num2

Mul | /mul/:num1/:num2

Div | /div/:num1/:num2

> Display division by zero not possible

2. It is not a REST API as http GET, PUT, POST are not required | WAIT NO—
3. So, we are using a custom HTTP Request
4. Getting num1 and num2 from the HTTP request
5. Computing the arithmetic and returning either a number or JSON object with body

```
{  
    Num1 :  
    Num2 :  
    Method : div  
    Output :  
    Comment : Division by zero not possible ! || Success !  
}
```


Or

3

Division by zero not possible

6. Storing the result is not required so no caching is required
7. Deploy 2 verticles 1 to get the input and 1 to produce and present the output
8. The verticle 1 should send the output (data) to verticle 2 through event bus

Quick setup — if you've done this kind of thing before

 Set up in Desktop or **HTTPS** **SSH**

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# calculator_vertx" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git branch -M main  
git remote add origin https://github.com/shreelakshmi-datakaveri/Calculator_vertx.git  
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/shreelakshmi-datakaveri/Calculator_vertx.git  
git branch -M main  
git push -u origin main
```

...or import code from another repository

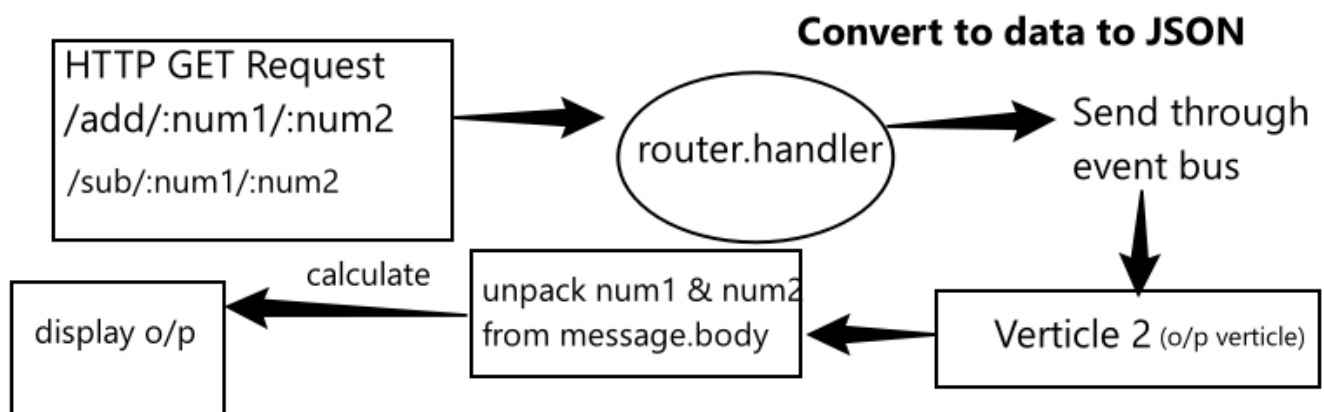
You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

 **ProTip!** Use the URL for this page when adding GitHub as a remote.

- The handler is asynchronously called when a client calls the GET endpoint
- As parameter to the handler, we get a routing context
- Routing context gives us access to request related information
- Returning a JSON array with multiple objects when the get endpoint is called ----->
- `context.response ()` creates a HTTP response , `.end()` we can pass a buffer or a string
- Now we have to use the newly created router and attach it to the HTTP server
- You can also run by using this command
`mvn clean compile vertx:run`
- Added the HTTP GET Endpoints
`/add/:num1/:num2`
`/sub/:num1/:num2`
`/mul/:num1/num2`
`/div/:num1/num2`
 If num2 is 0 -> It display : Cannot display by zero error
 Will also add a LOG.error output
- Implement what was told in 13th April google meet
- Convert the repeated list of lines into a method : DONE :)
- Deploy it on Docker too
- Add an error handling mechanism when the user enters different HTTP GET
 - Display : Input num1 and num2 in the HTTP request to add, subtract, multiply or divide
 - Also LOG.error
- Write Unit Test case
- EventBus with 2 verticle
- Promises
-

- Create a psvm + deploy the MainVerticle + error handlers + generate the project from vertx.io : **Done**
 - Set up the Logger : **Done**
 - JUnit Test cases : To check the code or flags
 - Deploy on Docker : **Done**
 - Set up router : **Done**
 - Set up 2 verticles : Main Verticle and output Verticle : **Done**
 - Set up event bus from the router | Check the type of the event Bus : **Done** || *But Couldn't send Data (Path Parameters through JSON as it's giving NULL error)*
 - Write some promises and Future : **Done**
 - Check how to use the browser along with Postman : **Done**
API Testing is remaining
 - Maybe add some more methods like calculating exponents, square roots, remainder
- De facto standard to convert the http request to JSON.toString and then send it via event bus



- Change the diagram because we are the output back to verticle 1 which displays HTTP response as a string
- Calculate and send the HTTP Response and display the output

Encountered with this error !

```
"C:\Program Files\Java\jdk-13.0.1\bin\java.exe" ...
14:46:28.492 INFO [vert.x-eventloop-thread-2] com.shree.calculator.vertx_starter.MainVerticle - Deployed class com.shree.calculator.vertx_starter.MainVerticle
14:46:28.492 INFO [vert.x-eventloop-thread-0] com.shree.calculator.vertx_starter.MainVerticle - HTTP server started on port 8888
14:46:28.730 ERROR [vert.x-eventloop-thread-1] com.shree.calculator.vertx_starter.MainVerticle - Unhandled : {}
java.lang.NumberFormatException: Create breakpoint : empty String <2 internal lines>
    at java.base/java.lang.Double.parseDouble(Double.java:549)
    at com.shree.calculator.vertx_starter.OutputVerticle.subtract(OutputVerticle.java:74)
    at com.shree.calculator.vertx_starter.OutputVerticle.lambda$start$1(OutputVerticle.java:20)
    at io.vertx.core.impl.EventLoopContext.emit(EventLoopContext.java:50)
    at io.vertx.core.impl.DuplicatedContext.emit(DuplicatedContext.java:168)
    at io.vertx.core.eventbus.impl.MessageConsumerImpl.dispatch(MessageConsumerImpl.java:177)
    at io.vertx.core.eventbus.impl.HandlerRegistration$InboundDeliveryContext.next(HandlerRegistration.java:169)
    at io.vertx.core.eventbus.impl.HandlerRegistration$InboundDeliveryContext.dispatch(HandlerRegistration.java:134)
    at io.vertx.core.impl.AbstractContext.dispatch(AbstractContext.java:111)
    at io.vertx.core.eventbus.impl.HandlerRegistration.dispatch(HandlerRegistration.java:105)
    at io.vertx.core.eventbus.impl.MessageConsumerImpl.deliver(MessageConsumerImpl.java:183)
    at io.vertx.core.eventbus.impl.MessageConsumerImpl.doReceive(MessageConsumerImpl.java:168)
    at io.vertx.core.eventbus.impl.HandlerRegistration.lambda$receive$0(HandlerRegistration.java:56)
    at io.netty.util.concurrent.AbstractEventExecutor.safeExecute(AbstractEventExecutor.java:164)
    at io.netty.util.concurrent.SingleThreadEventExecutor.runAllTasks(SingleThreadEventExecutor.java:469)
```

Even Though I have the same implementation for add and subtract !!

I think subtract and other methods are calling themselves when there is no HTTP request done and that's why the error !

Let me check

I'm a dummy! because I didn't send the message properly ! that's why message.body is empty

```
private void mul(RoutingContext routingContext) {
//    routingContext.response().end("Multiplying...");
    vertx.eventBus().request(MULTIPLY_ADDRESS, message: "", reply -> {
        routingContext.request().response().end((String) reply.result().body());
    });
}
```

```
private void sub(RoutingContext routingContext) {
//    routingContext.response().end("Subtracting...");
    String num1 = routingContext.pathParam(s: "num1");
    String num2 = routingContext.pathParam(s: "num2");
    String message = num1 + " " + num2;
    vertx.eventBus().request(SUBTRACT_ADDRESS, message, reply -> {
        routingContext.request().response().end((String) reply.result().body());
    });
}
```

But it should be this way

It's also funny that I didn't do any test driven development. In fact, I'm trying to write my test cases which are compatible with my API or HTTP endpoints !!!!!

- ❖ Vertx can be great for building APIs
- ❖ In the vertx world, verticles do all the work and they interact with each other using event Bus
- ❖ Future will return FAILURE or SUCCESS to the calling thread
- ❖ Futures are used to Handle the outcome of the promise

Output for promise_success in FuturePromiseExample class

```
"C:\Program Files\Java\jdk-13.0.1\bin\java.exe" ...
09:18:11.885 INFO [main] futureAndPromise.FuturePromiseExample - Start
09:18:11.892 INFO [main] futureAndPromise.FuturePromiseExample - End
09:18:12.392 INFO [vert.x-eventloop-thread-0] futureAndPromise.FuturePromiseExample - Success

Process finished with exit code 0
```

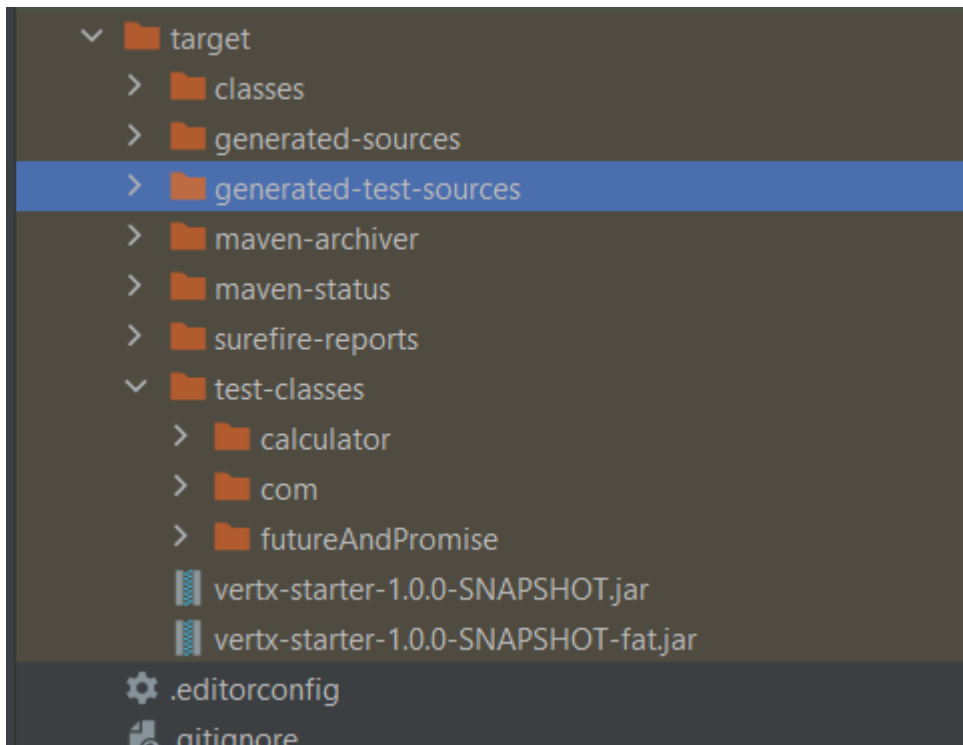
- As the timer is executed in the test instance (delay of 500 milliseconds), therefore End is printed before Success
 - This means the handler and the timer is executed asynchronously with a delay of half a second
 - When working with Asynchronous code, it is very important to know what handler is called first
 - Futures are used to Handle the outcome of the promise
-
- Due to shadow plugin the gradle or maven, when we build our application, a fat jar is created
 - Fat Jar is a single jar file which contains all the compiled java classes from our project and also all the compiled java classes from all the dependency
 - This means no additional dependencies has to be referenced
 - Fat jars are very handy when packaging the application inside a java container or for execution
 - We will now bundle fat jar by using Maven
- From <https://jenkov.com/tutorials/maven/maven-build-fat-jar.html>

The Maven command to make Maven build the Fat JAR for your project is:

```
mvn clean package
```

Output in the terminal :

```
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ vertx-starter ---
[INFO] Deleting C:\Users\shree\OneDrive\Documents\Project_Workspace\IUDX\practice\Calculator_vertx\target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ vertx-starter ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:compile (default-compile) @ vertx-starter ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 2 source files to C:\Users\shree\OneDrive\Documents\Project_Workspace\IUDX\practice\Calculator_vertx\target\classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ vertx-starter ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\Users\shree\OneDrive\Documents\Project_Workspace\IUDX\practice\Calculator_vertx\src\test\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:testCompile (default-testCompile) @ vertx-starter ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 3 source files to C:\Users\shree\OneDrive\Documents\Project_Workspace\IUDX\practice\Calculator_vertx\target\test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @ vertx-starter ---
```



Fat jar in the target directory

- Now, we will see to bundle fat jar into a docker container

```

=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/adoptopenjdk:11-jre-hotspot
=> [internal] load build context
=> => transferring context: 93B
=> [1/3] FROM docker.io/library/adoptopenjdk:11-jre-hotspot
=> CACHED [2/3] COPY target/vertx-starter-1.0.0-SNAPSHOT-fat.jar /usr/app/
=> CACHED [3/3] WORKDIR /usr/app
=> exporting to image
=> => exporting layers
=> => writing image sha256:11113482dd5f0ec3c6f937db08c96cae54af6f05564efcf0ac958d1b3f3a7da0
=> => naming to docker.io/example/vertx-starter

```

```

se 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
$ C:\Users\shree\OneDrive\Documents\Project_Workspace\IUDX\practice\Calculator_vertx> docker run -t -i -p 8888:8888 example/vertx-starter
5:48:16.898 INFO [vert.x-eventloop-thread-1] com.shree.calculator.vertx_starter.MainVerticle - HTTP server started on port 8888
5:48:16.899 INFO [vert.x-eventloop-thread-0] i.v.c.impl.launcher.commands.VertxIsolatedDeployer - Succeeded in deploying verticle

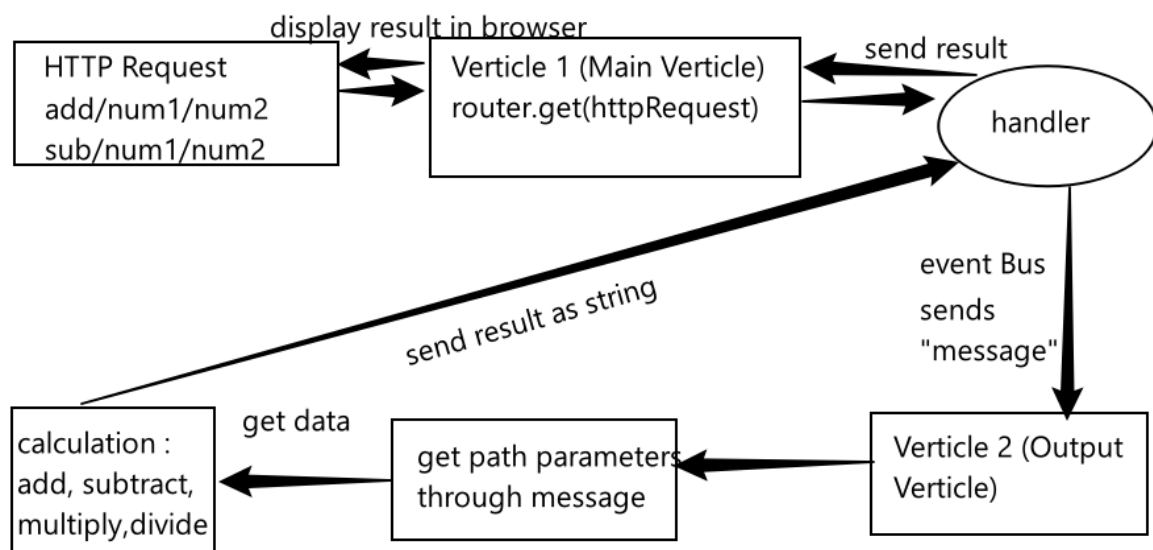
```

- So now, the main verticle can be run in the terminal by executing docker build and run commands
- Also, a new docker image is created and named as vertex - starter

Problem :

My git bash is configured to my previous git account and when I try to reconfigure it using my new git data kaveri account and then push the next commit, it added 2 contributors in GitHub !

That's why I deleted the whole repository and trying to do this all over again



But the data to be packaged in JSON format to send it to and from the event bus. But here, I'm sending data as String because JSON object seems to be NULL.

Added the above image in github issues to get this url

)