Ex No: 3

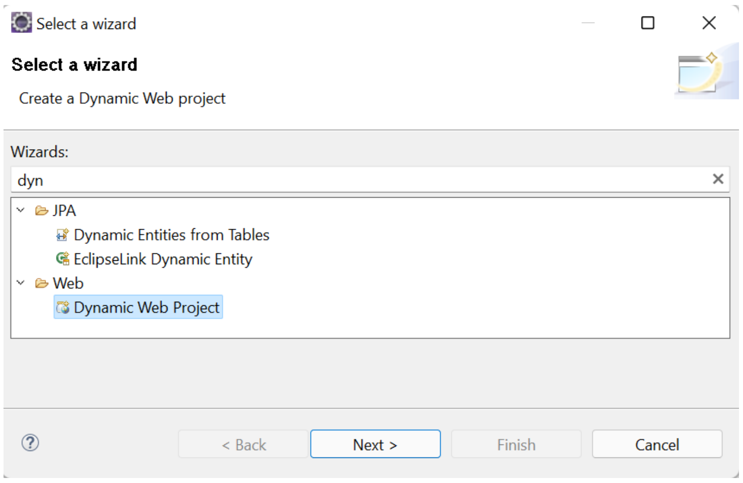
01/04/2022 **IMPLEMENTATION OF SOAP AND RESTFUL WEB SERVICE IN JAVA**

AIM:

To implement SOAP and REST using java.

SOAP:

Create a new dynamic web project in Eclipse IDE



Select a wizard

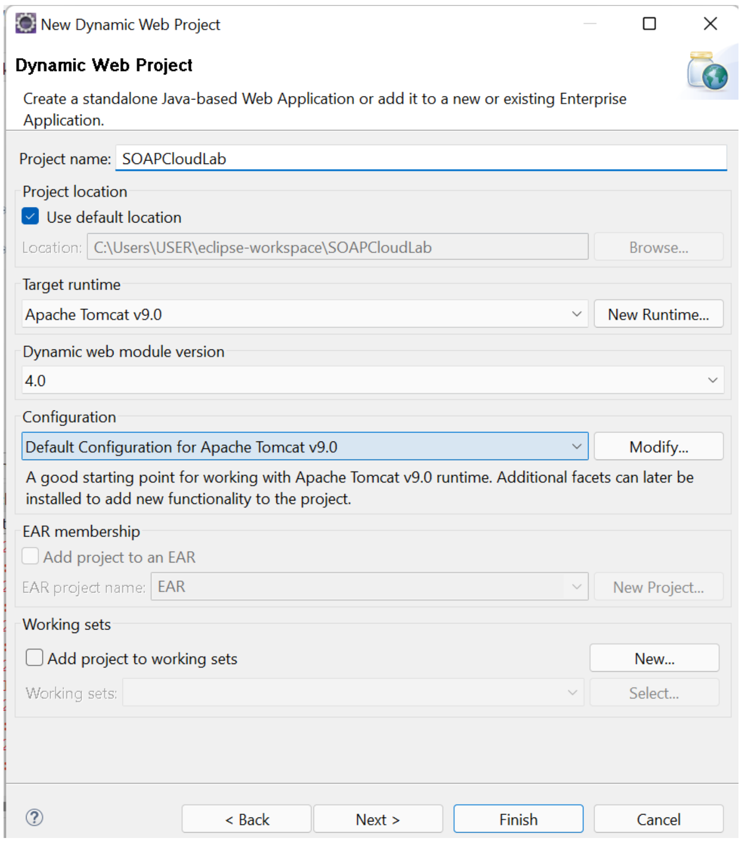
Select a wizard

Create a Dynamic Web project

Wizards: dyn

- JPA
  - Dynamic Entities from Tables
  - EclipseLink Dynamic Entity
- Web
  - Dynamic Web Project

Write a project name and check the server and runtime configuration and then click next and then finish.



New Dynamic Web Project

Dynamic Web Project

Create a standalone Java-based Web Application or add it to a new or existing Enterprise Application.

Project name: SOAPCloudLab

Project location

✓ Use default location

Location: C:\Users\USER\eclipse-workspace\SOAPCloudLab

Target runtime: Apache Tomcat v9.0

Dynamic web module version: 4.0

Configuration: Default Configuration for Apache Tomcat v9.0

A good starting point for working with Apache Tomcat v9.0 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

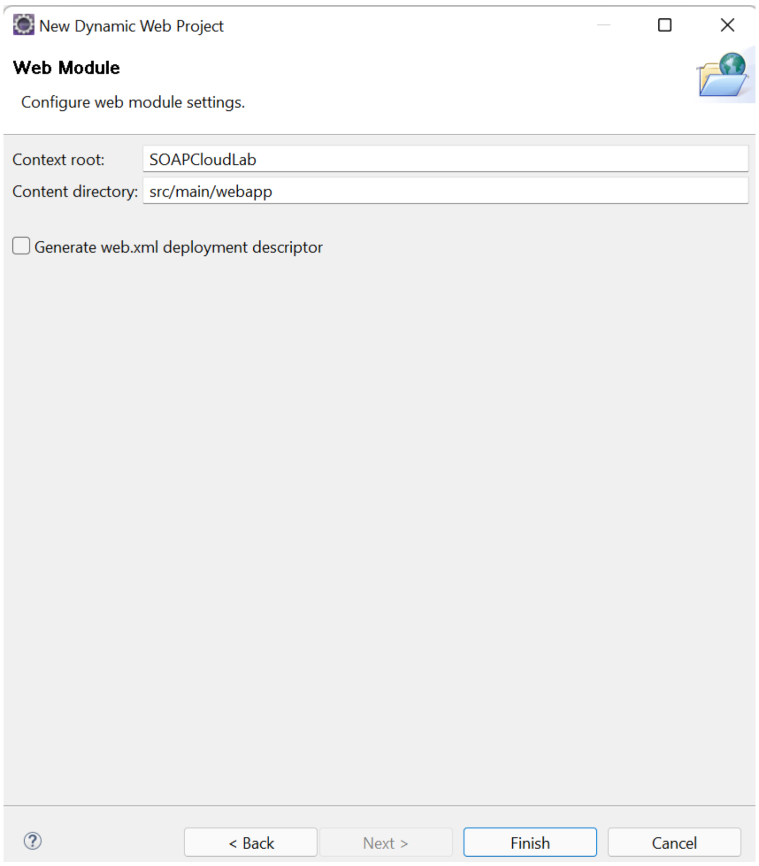
☐ Add project to an EAR

EAR project name: EAR

Working sets

☐ Add project to working sets

Working sets:



New Dynamic Web Project

Web Module

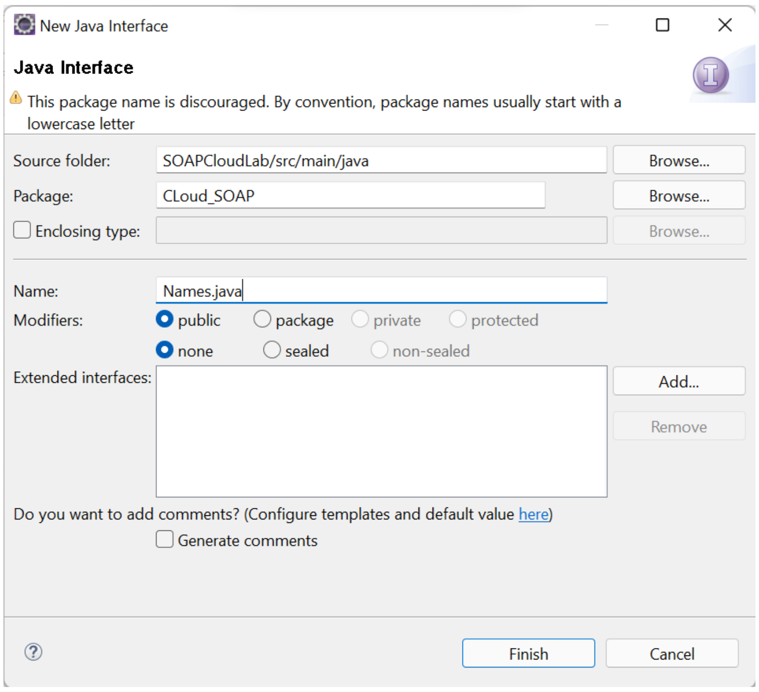
Configure web module settings.

Context root: SOAPCloudLab

Content directory: src/main/webapp

☐ Generate web.xml deployment descriptor

Create two files DisplayNames.java and Names.java



New Java Interface

Java Interface

This package name is discouraged. By convention, package names usually start with a lowercase letter

Source folder: SOAPCloudLab/src/main/java

Package: CLoud_SOAP

☐ Enclosing type:

Name: Names.java

Modifiers: [✓ public] package private protected

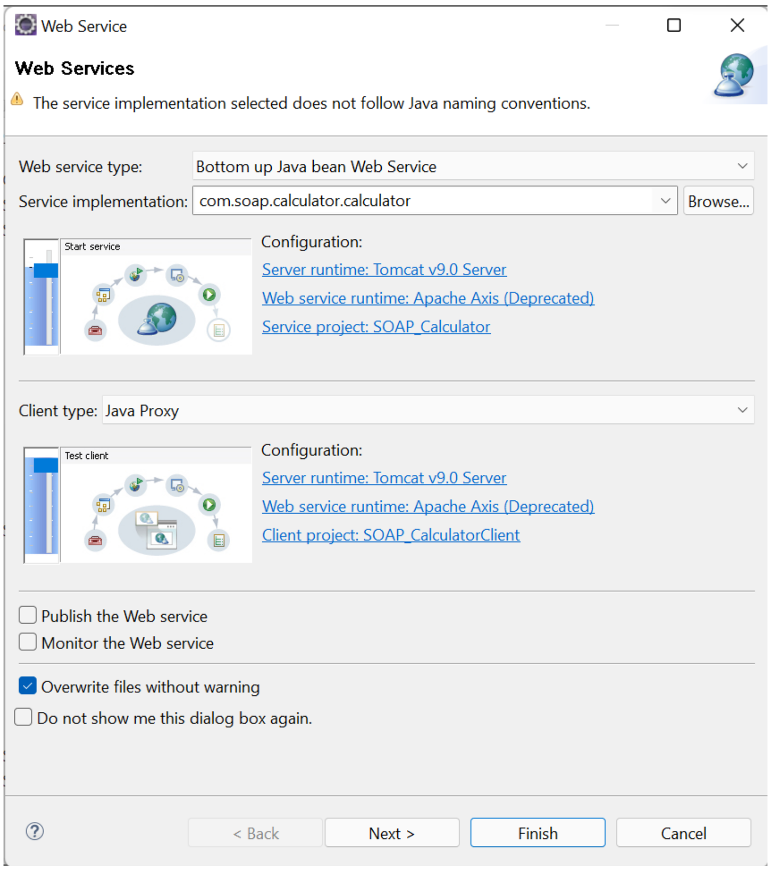
[✓ none] sealed non-sealed

Extended interfaces:

Do you want to add comments? (Configure templates and default value here)

☐ Generate comments

After entering the content of the files, right click on the files and click on web services.



Web Services

⚠ The service implementation selected does not follow Java naming conventions.

Web service type: Bottom up Java bean Web Service

Service implementation: com.soap.calculator.calculator

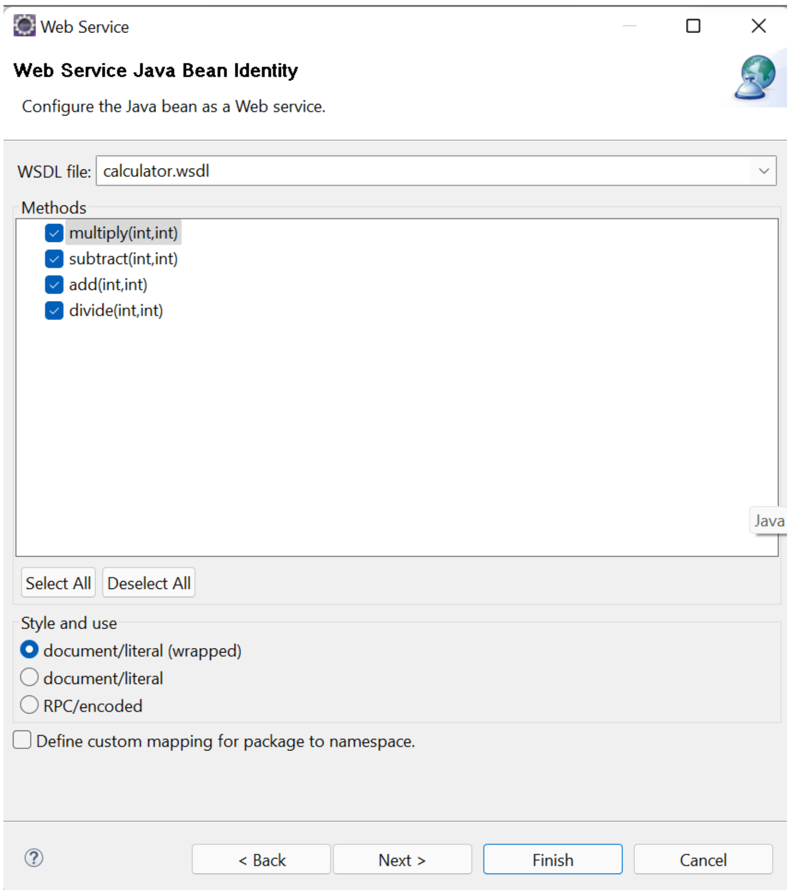
Configuration:
Server runtime: Tomcat v9.0 Server
Web service runtime: Apache Axis (Deprecated)
Service project: SOAP_Calculator

Client type: Java Proxy

Configuration:
Server runtime: Tomcat v9.0 Server
Web service runtime: Apache Axis (Deprecated)
Client project: SOAP_CalculatorClient

- [ ] Publish the Web service
- [ ] Monitor the Web service
- [✓] Overwrite files without warning
- [ ] Do not show me this dialog box again.

Click next



Web Service Java Bean Identity

Configure the Java bean as a Web service.

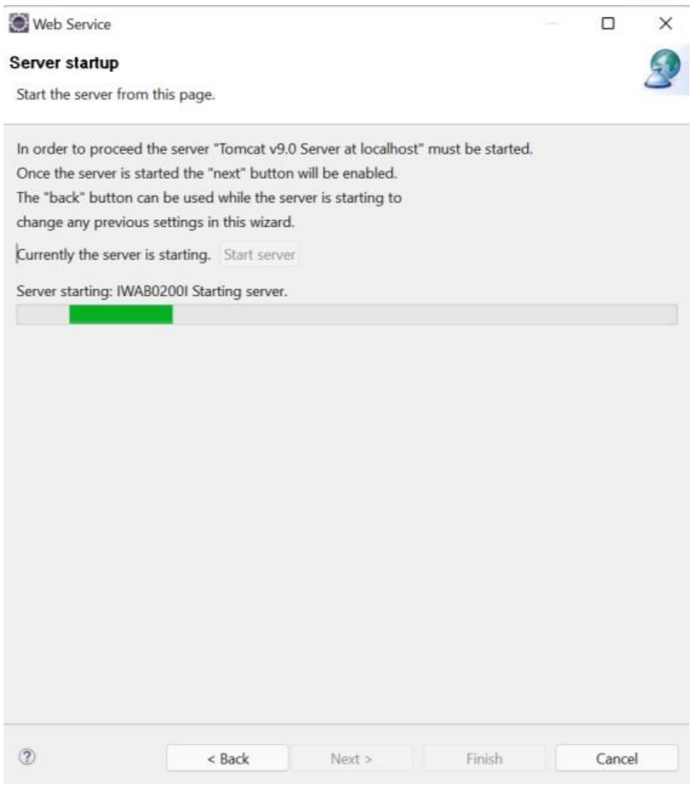
WSDL file: calculator.wsdl

Methods
- [✓] multiply(int,int)
- [✓] subtract(int,int)
- [✓] add(int,int)
- [✓] divide(int,int)

Style and use
- ● document/literal (wrapped)
- ○ document/literal
- ○ RPC/encoded

- [ ] Define custom mapping for package to namespace.

Click next and then start the server.

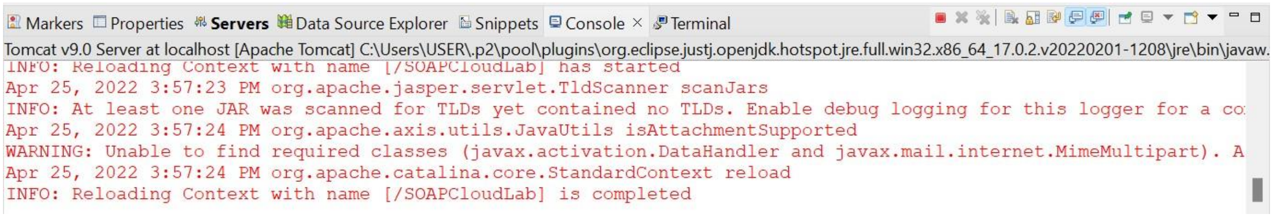


In order to proceed the server "Tomcat v9.0 Server at localhost" must be started.
Once the server is started the "next" button will be enabled.
The "back" button can be used while the server is starting to
change any previous settings in this wizard.

Currently the server is starting.

Server starting: IWAB0200I Starting server.

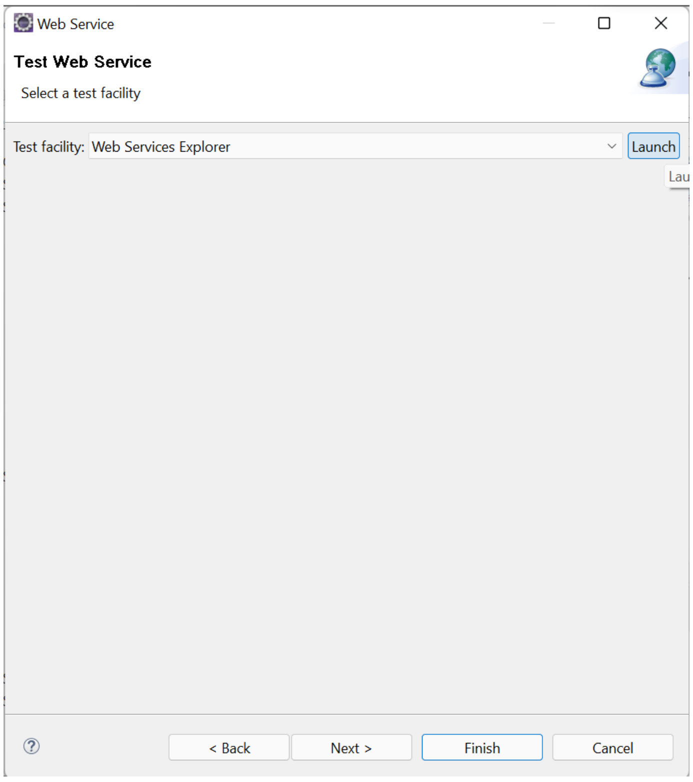
After server started successfully this message will displayed in the console.



Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Users\USER\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220201-1208\jre\bin\javaw.

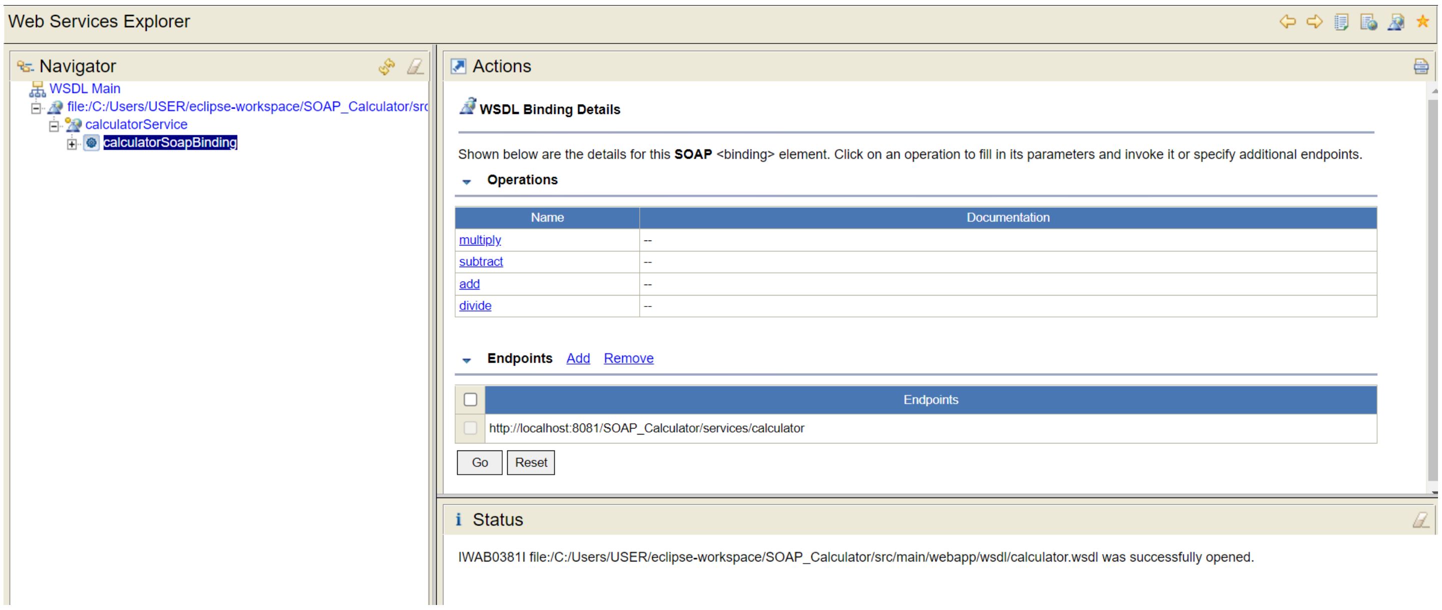
```
INFO: Reloading Context with name [/SOAPCloudLab] has started
Apr 25, 2022 3:57:23 PM org.apache.jasper.servlet.TldScanner scanJars
INFO: At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for this logger for a co
Apr 25, 2022 3:57:24 PM org.apache.axis.utils.JavaUtils isAttachmentSupported
WARNING: Unable to find required classes (javax.activation.DataHandler and javax.mail.internet.MimeMultipart). A
Apr 25, 2022 3:57:24 PM org.apache.catalina.core.StandardContext reload
INFO: Reloading Context with name [/SOAPCloudLab] is completed
```

After server started click on the Launch button



Test facility: Web Services Explorer

We will get web service explorer for our calculator application, after clicking on launch button.



Web Services Explorer

Navigator
WSDL Main
file:/C:/Users/USER/eclipse-workspace/SOAP_Calculator/src
calculatorService
calculatorSoapBinding

Actions

WSDL Binding Details

Shown below are the details for this **SOAP** <binding> element. Click on an operation to fill in its parameters and invoke it or specify additional endpoints.

Operations

Name	Documentation
multiply	--
subtract	--
add	--
divide	--

Endpoints Add Remove

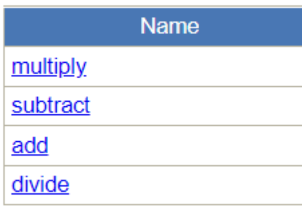
	Endpoints
☐	http://localhost:8081/SOAP_Calculator/services/calculator

Go Reset

Status

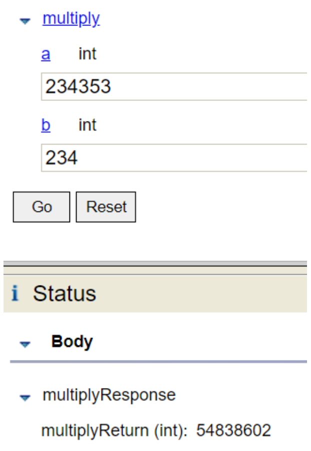
IWAB0381I file:/C:/Users/USER/eclipse-workspace/SOAP_Calculator/src/main/webapp/wsdl/calculator.wsdl was successfully opened.

In this web service explorer itself specified all the methods we have created.



Name
multiply
subtract
add
divide

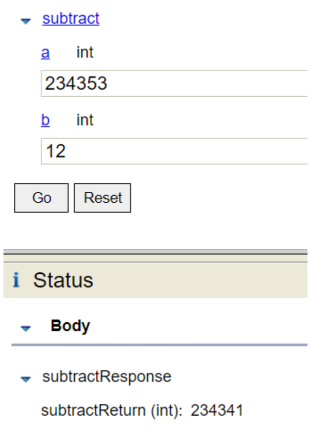
By clicking on the methods, we could able to do those operations.



multiply
a int
234353
b int
234
Go Reset

Status
Body
multiplyResponse
multiplyReturn (int): 54838602

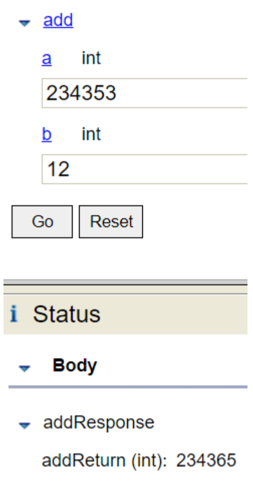
By clicking on multiply and by giving values for a and b and by clicking on Go, we can get the multiplication result.



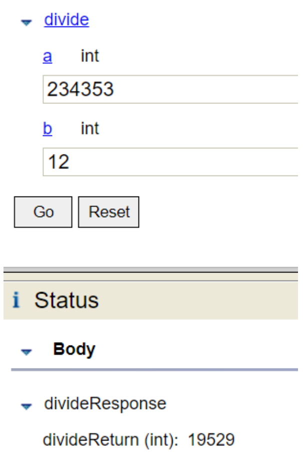
subtract
a int
234353
b int
12
Go Reset

Status
Body
subtractResponse
subtractReturn (int): 234341

By clicking on subtract and by giving values for a and b and by clicking on Go, we can get the subtraction result.


add
a int
234353
b int
12
Go Reset
Status
Body
addResponse
addReturn (int): 234365

By clicking on add and by giving values for a and b and by clicking on Go, we can get the addition result.


divide
a int
234353
b int
12
Go Reset
Status
Body
divideResponse
divideReturn (int): 19529

By clicking on divide and by giving values for a and b and by clicking on Go, we can get the division result.

CODE

calculator.java

```java
package com.soap.calculator;

public class calculator {
        public int add(int a, int b) {
            return (a + b);
        }

        public int subtract(int a, int b) {
            return (a - b);
        }

        public int multiply(int a, int b) {
            return (a * b);
        }

        public int divide(int a, int b) {
            return (a / b);
        }
}
```

Operations.java

```java
package com.soap.calculator;
```

```
public interface Operations {
     public int addition(int input1, int input2);
     public int subtraction(int input1, int input2);
     public int multiplication(int input1, int input2);
     public int division(int input1, int input2);
}
```

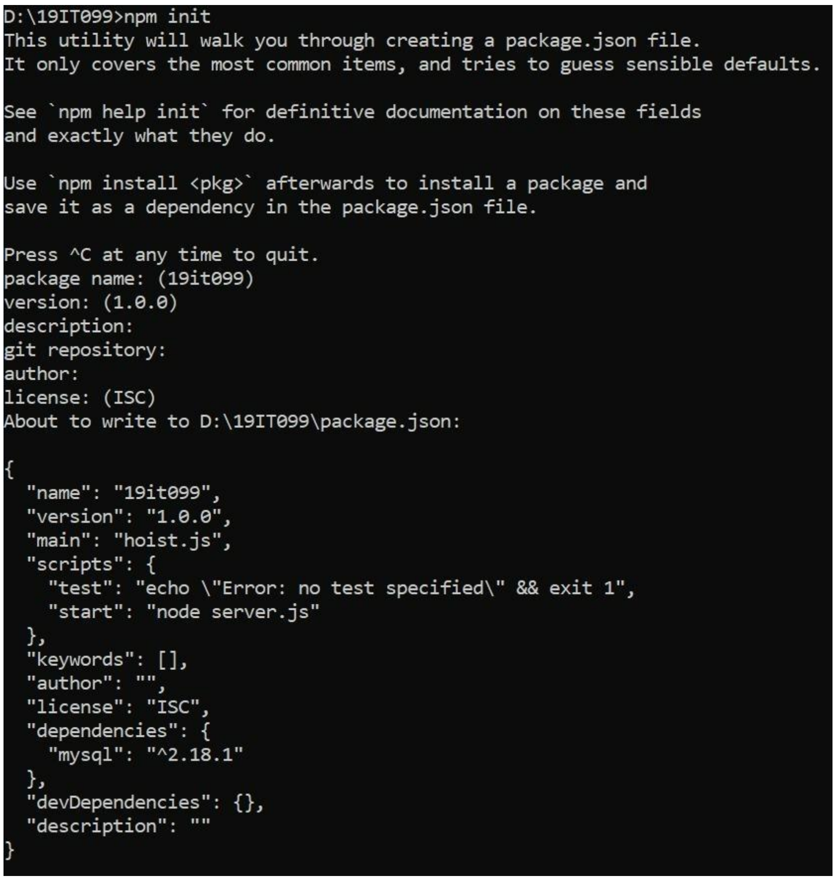
RESTFUL API IMPLEMENTATION

CODE

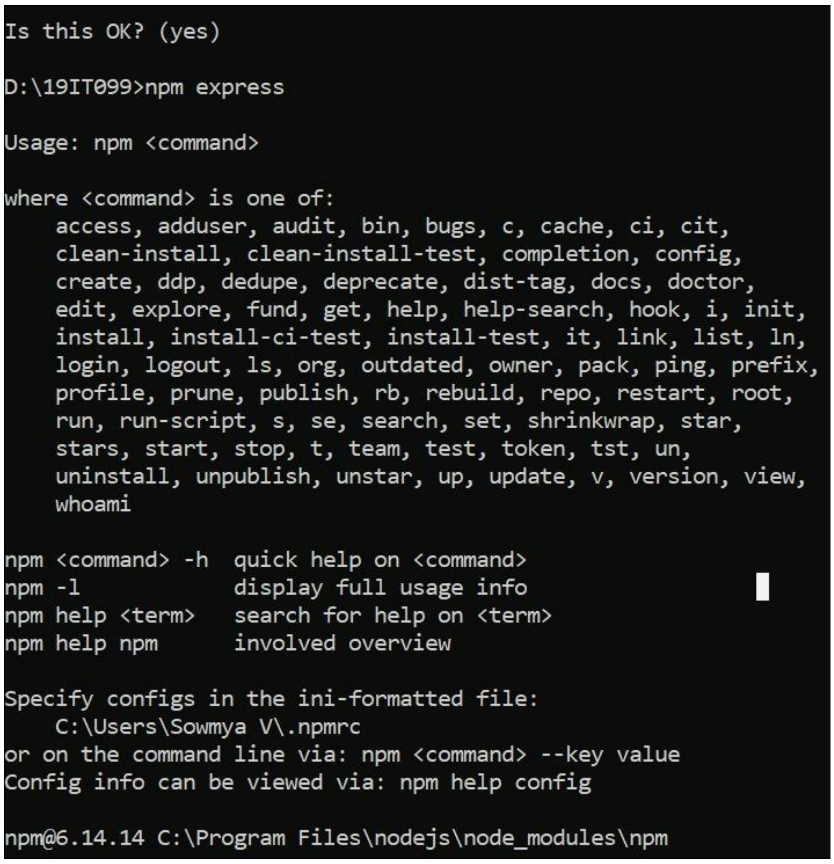
```
const express = require('express');
const Joi = require('joi'); //used for validation
const app = express();
app.use(express.json());
const customers = [
 { title: 'Sarvesh', id:84 },
 { title: 'Sowmya', id:99 },
 { title: 'Vibhisheak', id: 116 }
 ]
 //READ Request Handlers
app.get('/', (req, res) => {
 res.send('Welcome!!');
});
app.get('/api/customers', (req, res) => {
 res.send(customers);
});
app.get('/api/customers/:id', (req, res) => {
 const customer = customers.find(c => c.id === parseInt(req.params.id));
 if (!customer) res.status(404).send('<h2 style="font-family: Malgun Gothic; color: darkred;"></h2>');
 res.send(customer);
});
app.post('/api/customers', (req, res) => {

 const { error } = validatecustomer(req.body);
 if (error) {
 res.status(400).send(error.details[0].message)
 return;
 }
 const customer = {
```

```
 id: customers.length + 1,
 title: req.body.title
 };
 customers.push(customer);
 res.send(customer);
});
//UPDATE Request Handler
app.put('/api/customers/:id', (req, res) => {
 const customer = customers.find(c => c.id === parseInt(req.params.id));
 if (!customer) res.status(404).send(
 '<h2 style="font-family: Malgun Gothic; color: darkred;">Not Found!! </h2>');
 const { error } = validatecustomer(req.body);
 if (error) {
 res.status(400).send(error.details[0].message);
 return;
 }
 customer.title = req.body.title;
 res.send(customer);
});
//DELETE Request Handler
app.delete('/api/customers/:id', (req, res) => {
 const customer = customers.find(c => c.id === parseInt(req.params.id));
 if (!customer) res.status(404).send('<h2 style="font-family: Malgun Gothic; color: darkred;"> Not Found!! <
/h2>');
 const index = customers.indexOf(customer);
 customers.splice(index, 1);
 res.send(customer);
});
function validatecustomer(customer) {
 const schema = {
 title: Joi.string().min(3).required()
 };
 return Joi.validate(customer, schema);
}
const port = process.env.PORT || 8080;
app.listen(port, () => console.log('Listening on port ${port}..'));
```

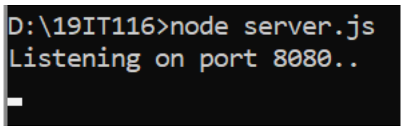
OUTPUT



```
D:\19IT099>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (19it099)
version: (1.0.0)
description:
git repository:
author:
license: (ISC)
About to write to D:\19IT099\package.json:

{
  "name": "19it099",
  "version": "1.0.0",
  "main": "hoist.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "mysql": "^2.18.1"
  },
  "devDependencies": {},
  "description": ""
}
```

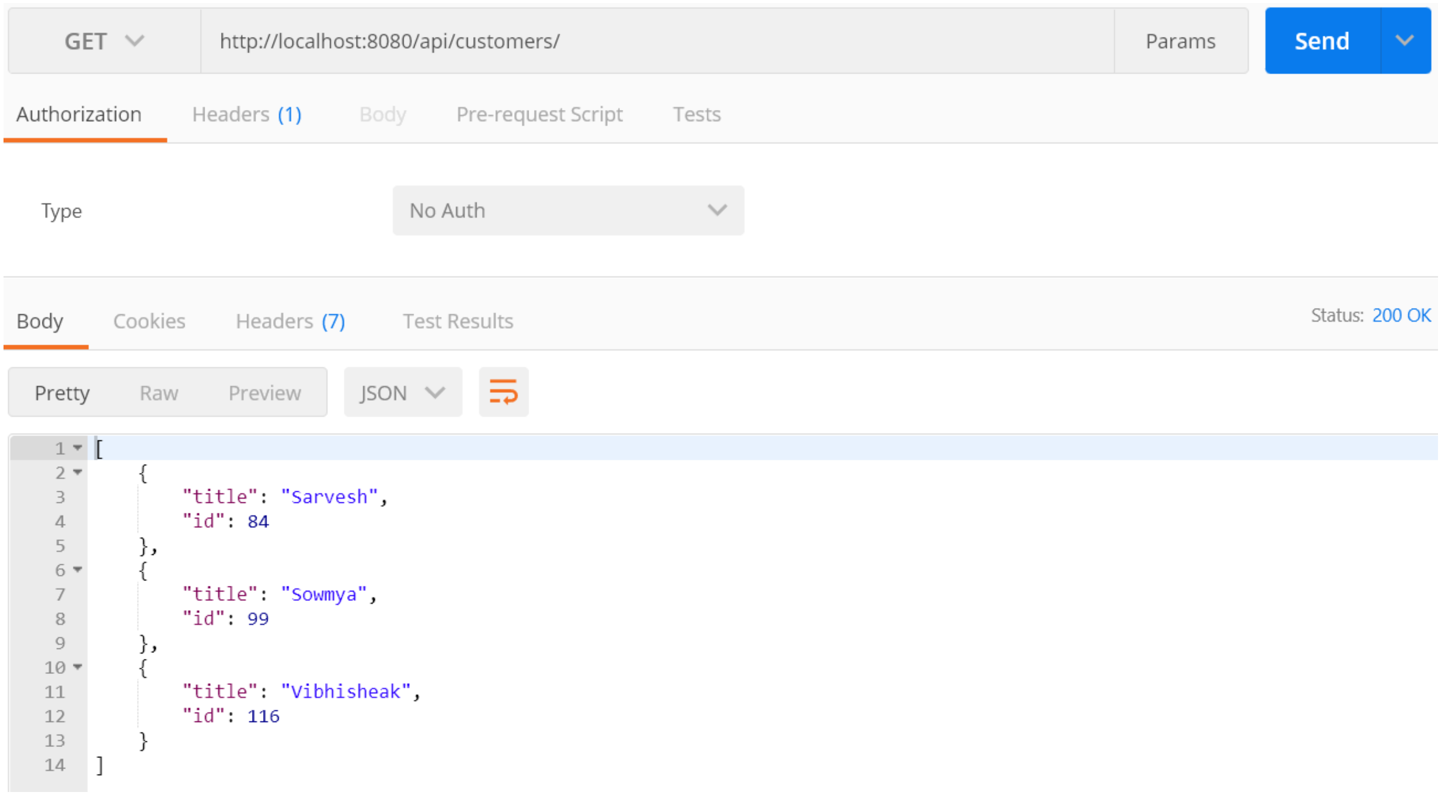


```
Is this OK? (yes)

D:\19IT099>npm express

Usage: npm <command>

where <command> is one of:
    access, adduser, audit, bin, bugs, c, cache, ci, cit,
    clean-install, clean-install-test, completion, config,
    create, ddp, dedupe, deprecate, dist-tag, docs, doctor,
    edit, explore, fund, get, help, help-search, hook, i, init,
    install, install-ci-test, install-test, it, link, list, ln,
    login, logout, ls, org, outdated, owner, pack, ping, prefix,
    profile, prune, publish, rb, rebuild, repo, restart, root,
    run, run-script, s, se, search, set, shrinkwrap, star,
    stars, start, stop, t, team, test, token, tst, un,
    uninstall, unpublish, unstar, up, update, v, version, view,
    whoami

npm <command> -h  quick help on <command>
npm -l            display full usage info
npm help <term>   search for help on <term>
npm help npm      involved overview

Specify configs in the ini-formatted file:
    C:\Users\Sowmya V\.npmrc
or on the command line via: npm <command> --key value
Config info can be viewed via: npm help config

npm@6.14.14 C:\Program Files\nodejs\node_modules\npm
```



```
D:\19IT116>node server.js
Listening on port 8080..
```

After starting the server, we have to open postman tool to do CRUD operations.
We could able to get all the customers details by giving the URL and by selecting GET and by sending it.



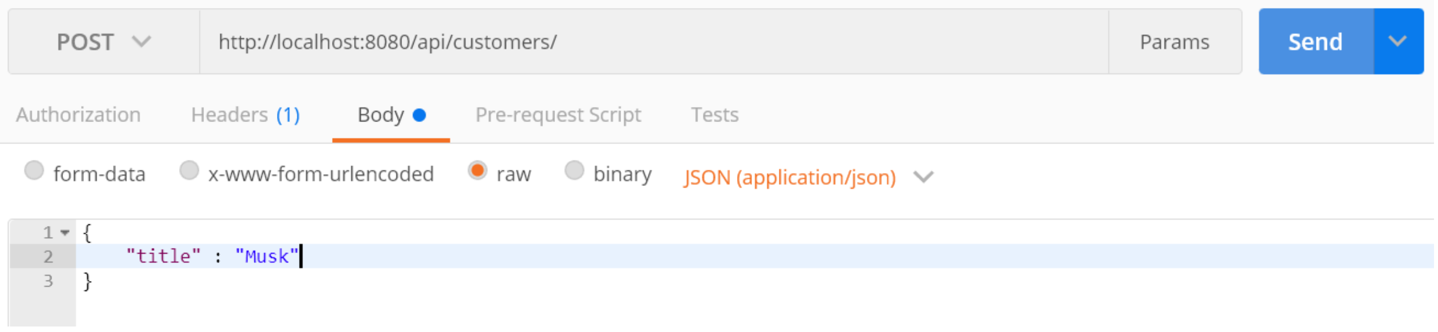
http://localhost:8080/api/customers/

Type: No Auth

Status: 200 OK

```
[
    {
        "title": "Sarvesh",
        "id": 84
    },
    {
        "title": "Sowmya",
        "id": 99
    },
    {
        "title": "Vibhisheak",
        "id": 116
    }
]
```

By posting the customer name, we could able to add new customer with new ID.

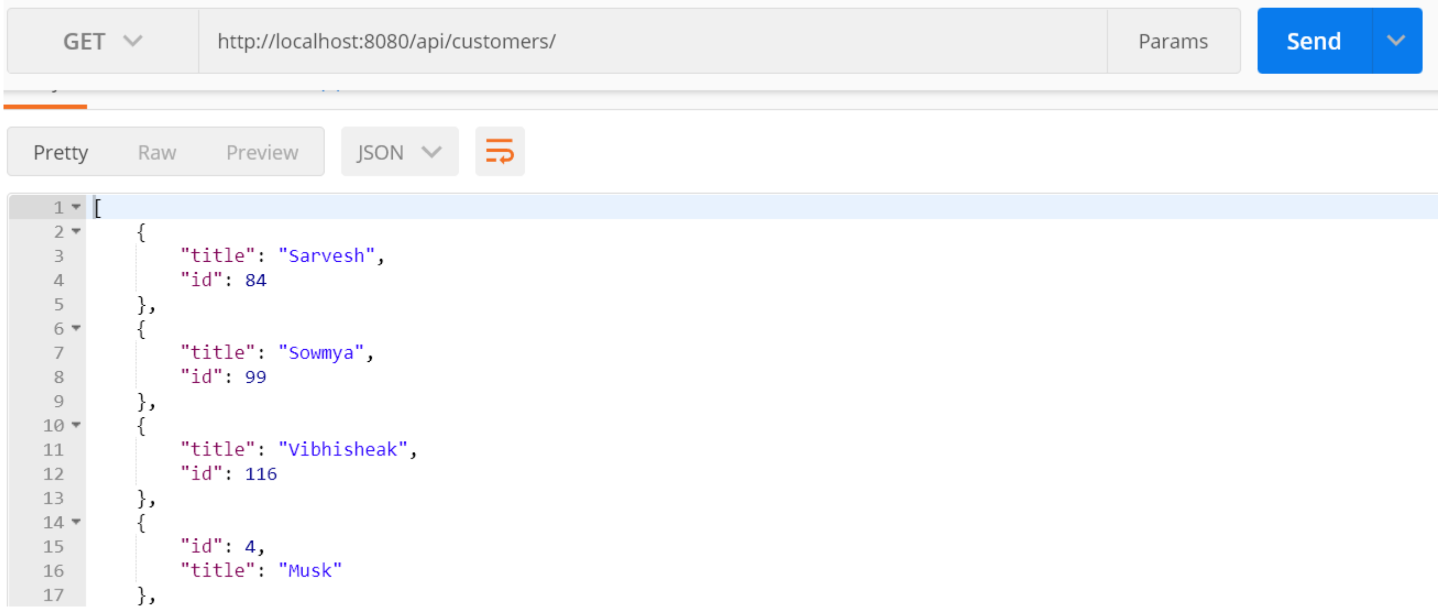


http://localhost:8080/api/customers/

JSON (application/json)

```
{
    "title" : "Musk"
}
```

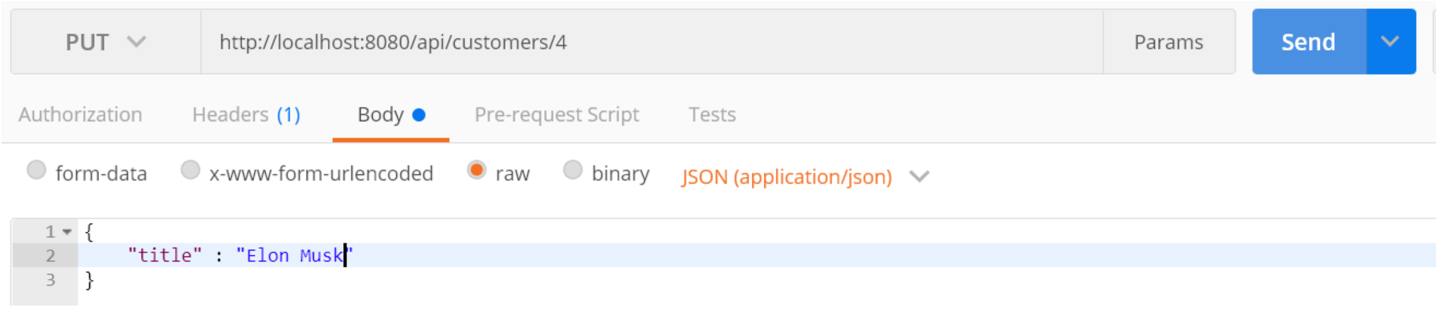
By clicking on get we could able to see the newly added customer details. Here musk has been added with new id 4.



http://localhost:8080/api/customers/

```
[
    {
        "title": "Sarvesh",
        "id": 84
    },
    {
        "title": "Sowmya",
        "id": 99
    },
    {
        "title": "Vibhisheak",
        "id": 116
    },
    {
        "id": 4,
        "title": "Musk"
    },
```

By selecting PUT, we could able to edit the details of the particular customers by giving their ID in the URL part.

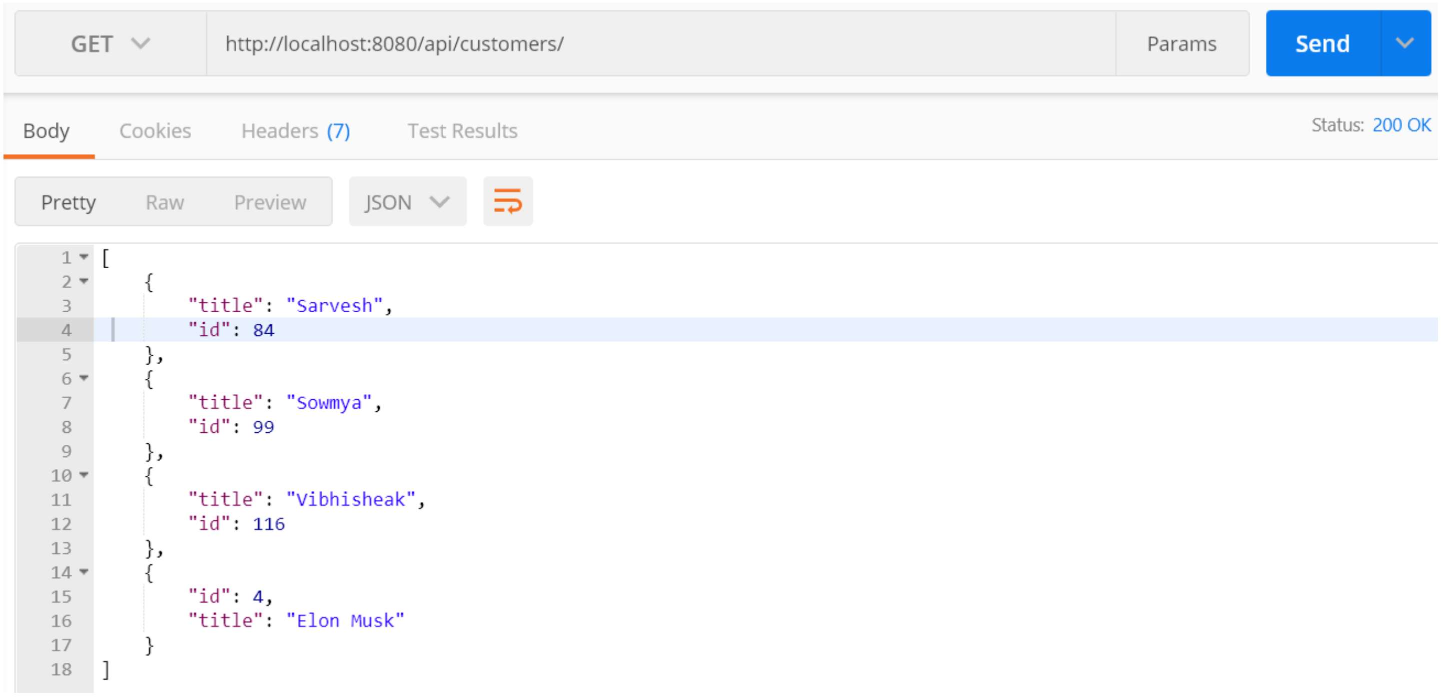


http://localhost:8080/api/customers/4

JSON (application/json)

```
{
    "title" : "Elon Musk"
}
```

By selecting GET we could able to see the altered details of Musk.

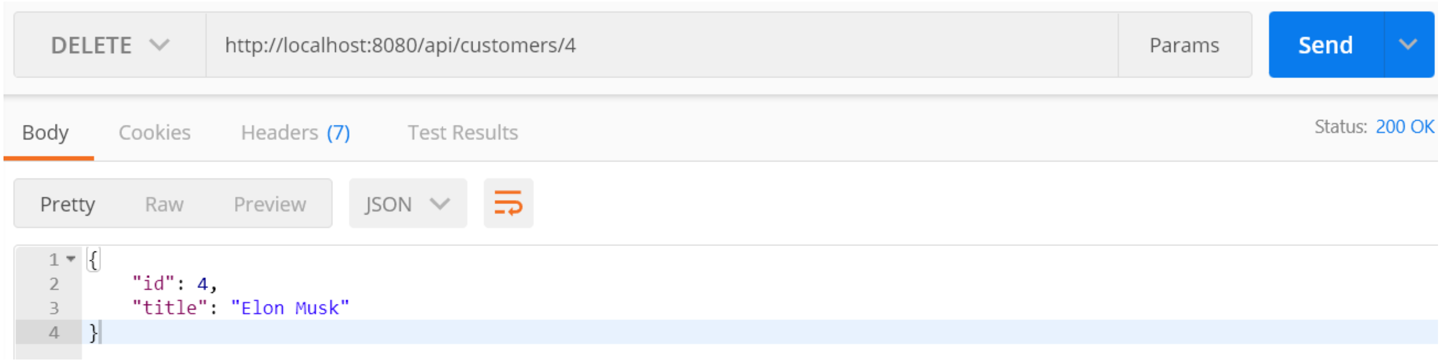


http://localhost:8080/api/customers/

Status: 200 OK

```
[
    {
        "title": "Sarvesh",
        "id": 84
    },
    {
        "title": "Sowmya",
        "id": 99
    },
    {
        "title": "Vibhisheak",
        "id": 116
    },
    {
        "id": 4,
        "title": "Elon Musk"
    }
]
```

For deleting the particular user, we have to give the respective ID in URL part and click on delete will delete the customer details completely.

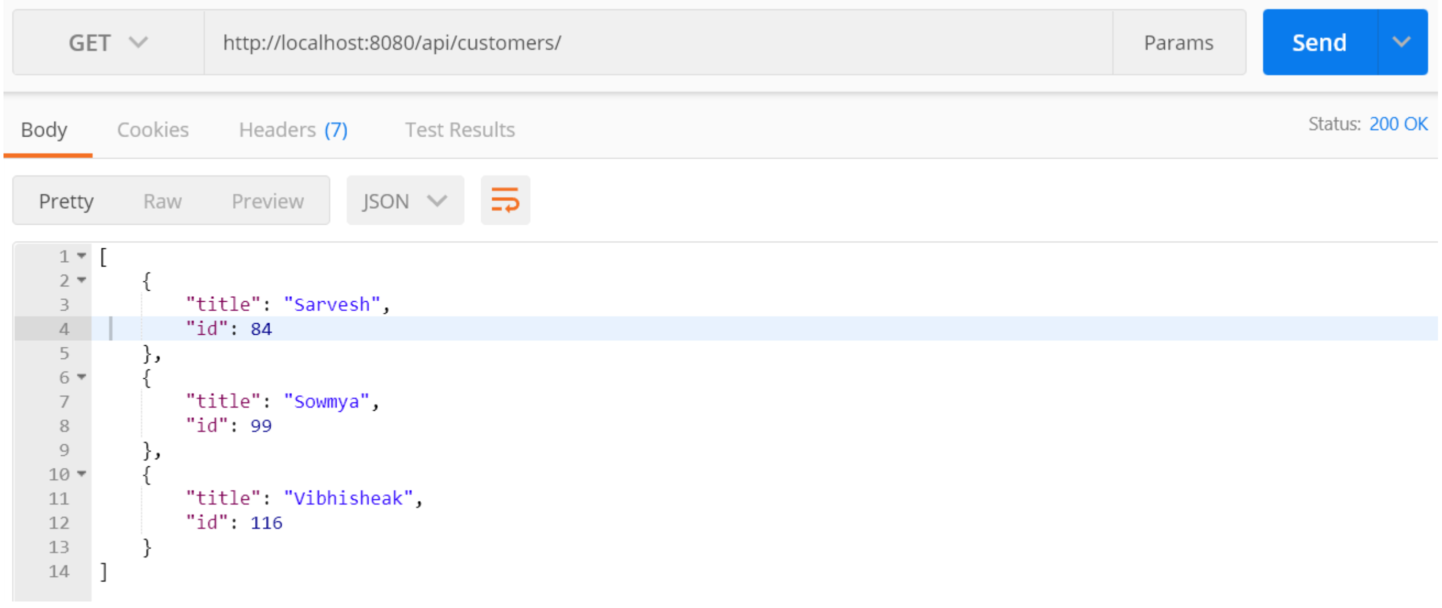


http://localhost:8080/api/customers/4

Status: 200 OK

```
{
    "id": 4,
    "title": "Elon Musk"
}
```

By selecting GET we could able to updated details about customers



http://localhost:8080/api/customers/

Status: 200 OK

```
[
    {
        "title": "Sarvesh",
        "id": 84
    },
    {
        "title": "Sowmya",
        "id": 99
    },
    {
        "title": "Vibhisheak",
        "id": 116
    }
]
```

RESULT

Thus, the installation and configuration of SOAP and restful web service implemented successfully.