Web Application Framework

For Developing the Web Applications

Web Application Frameworks are server-side software frameworks designed to enhance the development, maintenance, and scalability of web applications. These frameworks offer a suite of tools and libraries that streamline various web development tasks. In the context of web communication, servers and browsers interact via the HTTP protocol, with servers responding to HTTP requests from browsers.

Web Application Frameworks provide the advantage of employing simplified syntax, which automatically generates server-side code to handle these requests and responses. Consequently, developers benefit from working with higher-level, more accessible code, rather than getting involved in intricate low-level networking primitives. By leveraging these frameworks, developers can streamline their workflow and focus on the core functionality and features of their web applications.

A java-based web services framework, which provide an easier way to setup and create backend/server side of the web application.

Benefit of using this framework.

No need to configure XML file (deployment descriptor).
No need to write servlets classes.
No need to worry about GET/POST request and how to handle them
No need to worry about Forwarding or Scopes and how to handle them
No need to worry about Response headers and Secured guard and how to handle
You just need to declare all Annotation are present
How to use this framework

Setting up

Copy tmwebrockFramework.jar ,gson-2.10.1.jar file

and Paste the jar file on

tomcat9/webapps/PROJECT-NAME/WEB-INF/lib/.

Then

Copy/cut web.xml to tomcat9/Webapps/PROJECT-NAME/WEB-INF/.

Do some Changes in web.xml file.

You need to specify param-value against param-name 'SERVICE_PACKAGE_PREFIX'. In this project it is 'bobby.com.thinking.machines'.

User just need to change/write a single word inside web.xml and that was the value of param-name 'SERVICE_PACKAGE_PREFIX' by default there was "bobby.com.thinking.machines", user have to change it.

According to there folder structure

Note : The folder-name mentioned should exists inside tomcat9/Webapps/"project name"/WEB-INF/classes/.

```
<context-param>
<param-name>SERVICE_PACKAGE_PREFIX</param-name>
<param-value>bobby.com.thinking.machines</param-value>
</context-param>
```

Web.xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                      http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
  version="4.0"
  metadata-complete="true">

  <display-name>TMWebRock</display-name>
  <description>
Webservices Framework
  </description>
  <request-character-encoding>UTF-8</request-character-encoding>


<listener>
<listener-class>com.thinking.machines.webrock.TMWebRockStarter</listener-class>
</listener>

<context-param>
<param-name>SERVICE_PACKAGE_PREFIX</param-name>
<param-value>bobby.com.thinking.machines</param-value>
</context-param>
<servlet>
<servlet-name>TMWebRock</servlet-name>
```

```
<servlet-class>com.thinking.machines.webrock.TMWebRock</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>TMWebRock</servlet-name>
<url-pattern>/webServices/*</url-pattern>
</servlet-mapping>
</web-app>
```

User also have to change a single word inside web.xml, instead of 'param-value' user have to write application path name there.

Now copy all files inside Dependencies folder and paste them inside tomcat9/Webapps/"project name"/WEB-INF/lib/.

Our framework is Dependent on some of the these files.

You are done setting up the environment, now you can use the framework easily.

@Path Annotations:

Path annotation can be applied on class and method. The value of this annotation should always starts with front slash (/) followed by path.(By default method type is Get)

```
@Path("/sam")
public class Sam
{
@Path("/get")
public void get()
{
System.out.println("Get service got called");
}
}
```

user can access this service by sending request to "User's entity name"/employee/view.
Example url to access add service : http://localhost:8080/user-applicationcontext/webServices/employee/view

Let’s explore more annotation’s
In WEB-INF\classes\bobby\com\thinking\machines

```
class Student
{// i can also go for private property
public int rollNumber;
public String name;
public Student(int rollNumber,String name)
{
this.rollNumber=rollNumber;
this.name=name;
}
//Setter Getter
}
```

@RequestParameter Annotation

@Post Annotation for Post type Request

@Get Annotation for Get type request

can only be applied on Parameter. User can use the following annotation to request data which arrives in the web request. Search for data with given entity in request Bag and provide if found. User is supposed to apply Request Parameter annotation on all the primitive types.

If you don’t want to return anything then you can go for void & if you want to return any type of matter then you can go for Object
Object will be converted to json string appropriate object with result format.
We can handle json string as argument or return type.

You can only <u>use Complex data type as first parameter</u>

[Example are mostly is tested in Postman]

Example --

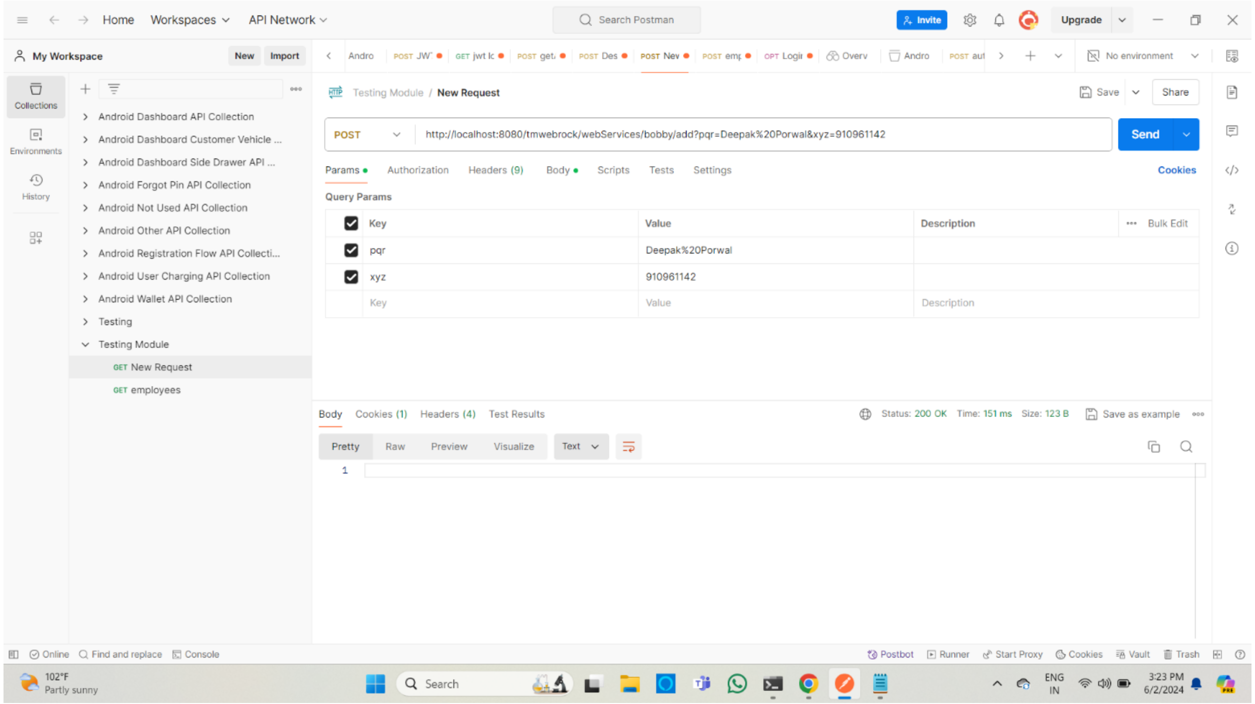
```
package bobby.com.thinking.machines;
import com.thinking.machines.webrock.annotations.*;
import com.thinking.machines.webrock.*;
import java.util.*;
```

```
@Path("/bobby")
public class BobbyService
{
@Post
@Path("/add")
public void add(@RequestParameter("pqr") String
name,@RequestParameter("xyz") long mobileNumber)
{
System.out.println("Name : "+name);
System.out.println("Mobile Number : "+mobileNumber);
System.out.println("add got called Bobby Add (Module) got called");
}

@Get
@Path("/get")
public Object  get()
{
System.out.println("get got called from Bobby Service");
String d="Ujjain is the city of god we live in Ujjain";
Student student=new Student(100,"Deepak Porwal");
return student;
}

@Get
@Path("/getAll")
public Object getAll()
{
System.out.println("getAll got called from bobby Service");
Student s1=new Student(100,"Mohit");
Student s2=new Student(200,"Rajveer");
Student s3=new Student(300,"Tom");
Student s4=new Student(400,"Jack");
List<Student> lists=new LinkedList<>();
lists.add(s1);
lists.add(s2);
lists.add(s3);
lists.add(s4);
return lists;
}
@Post
@Path("/addStudent")
public Object addStudent(Student student)
{
System.out.println("addStudent got called");
System.out.println("RollNumber : "+student.rollNumber);
System.out.println("Name : "+student.name);
return 1004;// means successfully added
}

}
```

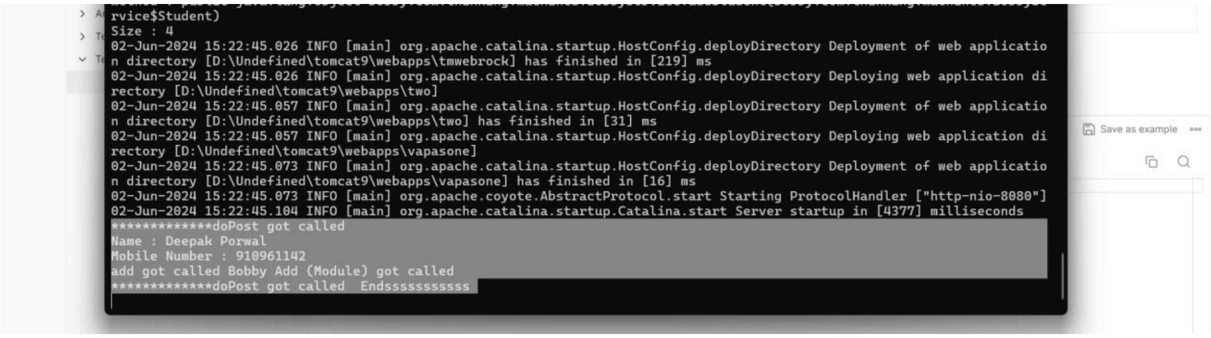


POST http://localhost:8080/tmwebrock/webServices/bobby/add?pqr=Deepak%20Porwal&xyz=910961142

Query Params

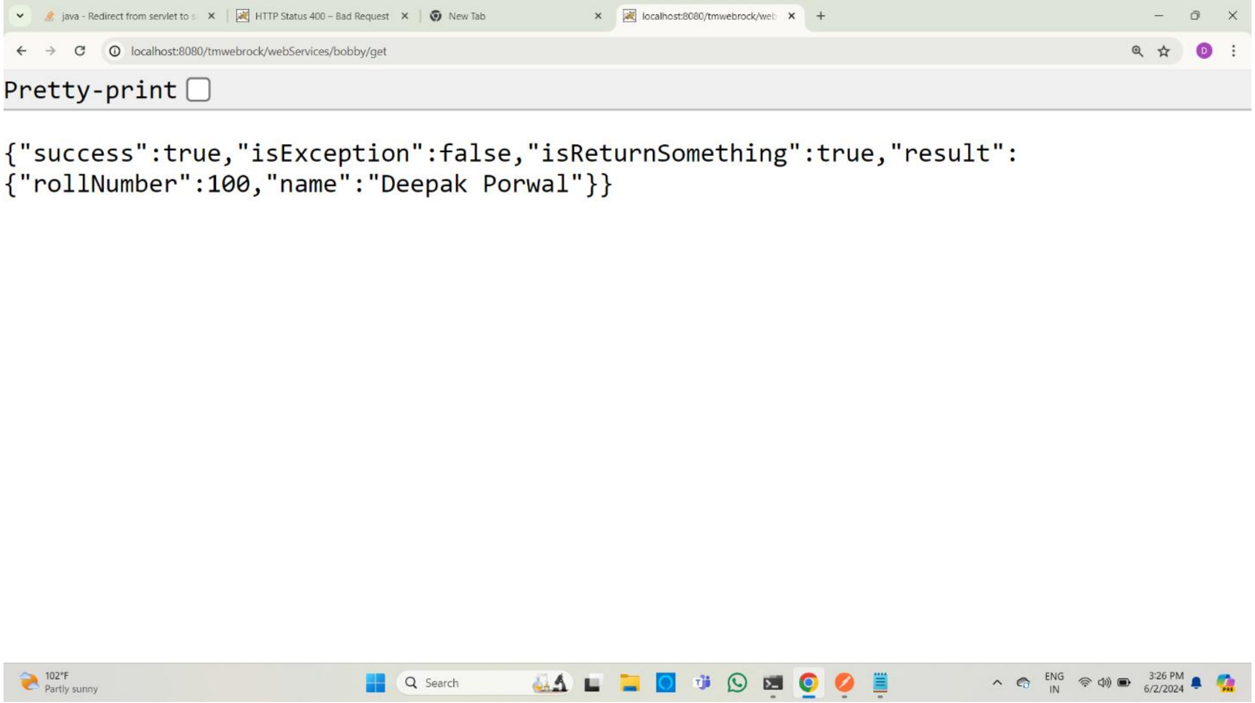
☑	Key	Value	Description
☑	pqr	Deepak%20Porwal	
☑	xyz	910961142	
	Key	Value	Description

Status: 200 OK Time: 151 ms Size: 123 B



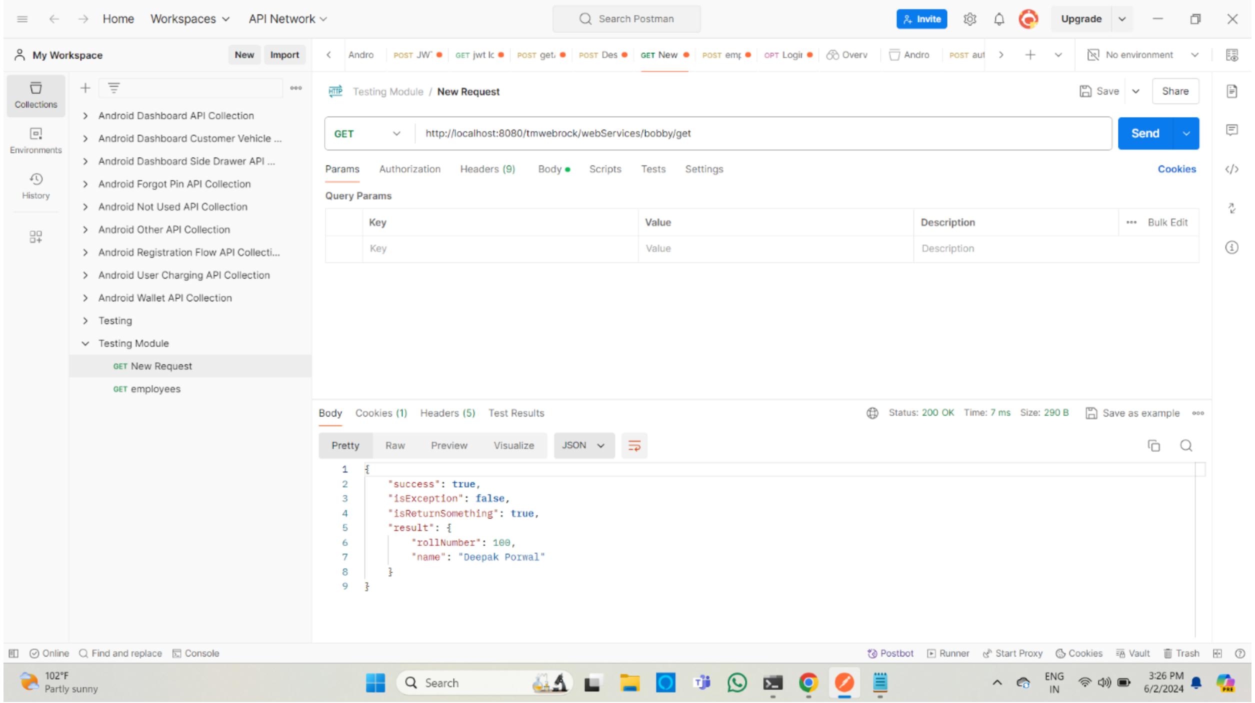
```
rvice$Student)
Size : 4
02-Jun-2024 15:22:45.026 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web applicatio
n directory [D:\Undefined\tomcat9\webapps\tmwebrock] has finished in [219] ms
02-Jun-2024 15:22:45.026 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application di
rectory [D:\Undefined\tomcat9\webapps\two]
02-Jun-2024 15:22:45.057 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web applicatio
n directory [D:\Undefined\tomcat9\webapps\two] has finished in [31] ms
02-Jun-2024 15:22:45.057 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application di
rectory [D:\Undefined\tomcat9\webapps\vapasone]
02-Jun-2024 15:22:45.073 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web applicatio
n directory [D:\Undefined\tomcat9\webapps\vapasone] has finished in [16] ms
02-Jun-2024 15:22:45.073 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8080"]
02-Jun-2024 15:22:45.104 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in [4377] milliseconds
*************doPost got called
Name : Deepak Porwal
Mobile Number : 910961142
add got called Bobby Add (Module) got called
*************doPost got called  Endsssssssssss
```

Another service getAll


java - Redirect from servlet to s
HTTP Status 400 – Bad Request
New Tab
localhost:8080/tmwebrock/web
Pretty-print □
102°F
Partly sunny
Search
ENG
IN
3:26 PM
6/2/2024

localhost:8080/tmwebrock/webServices/bobby/get

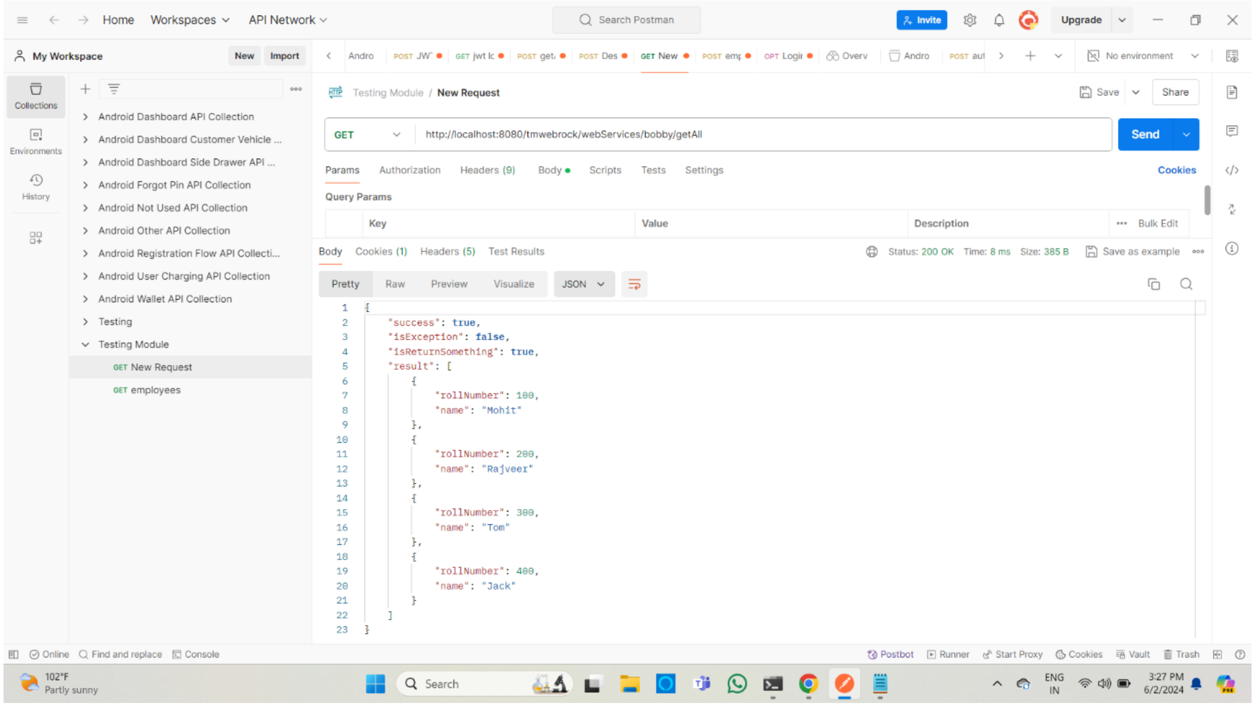
```
{"success":true,"isException":false,"isReturnSomething":true,"result":
{"rollNumber":100,"name":"Deepak Porwal"}}
```



GET http://localhost:8080/tmwebrock/webServices/bobby/get

Status: 200 OK Time: 7 ms Size: 290 B

```
{
    "success": true,
    "isException": false,
    "isReturnSomething": true,
    "result": {
        "rollNumber": 100,
        "name": "Deepak Porwal"
    }
}
```

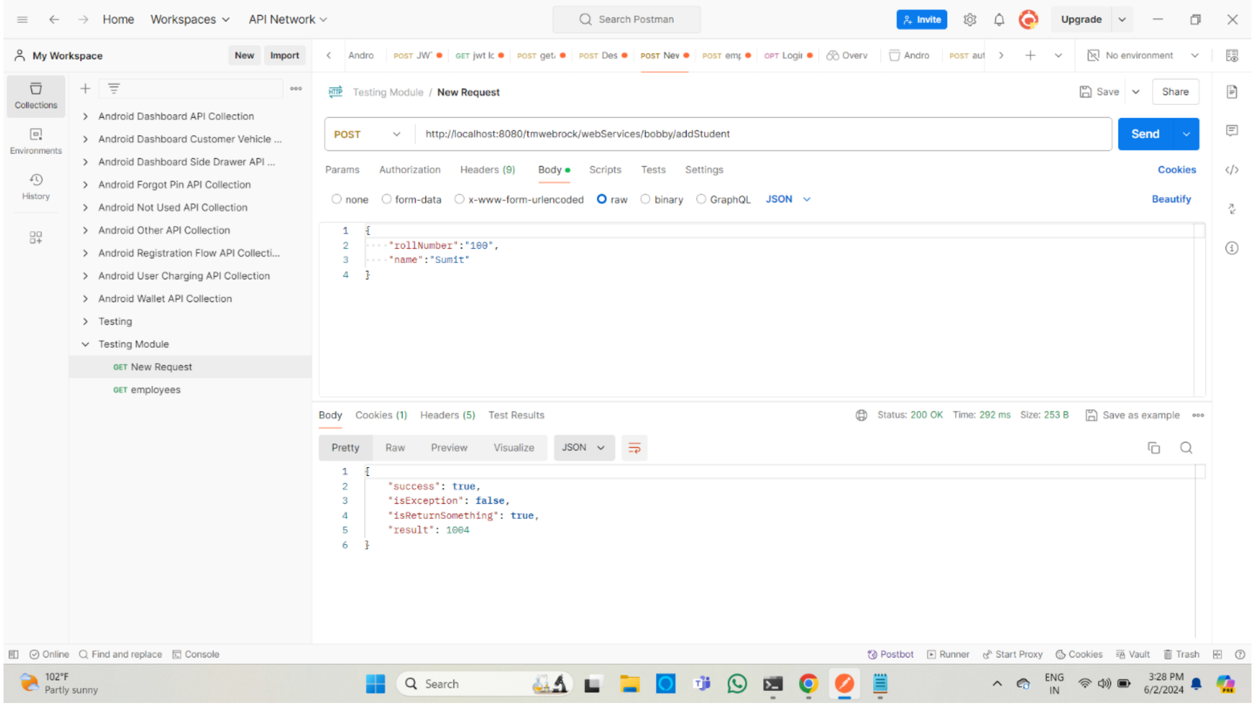


Testing Module / New Request

GET http://localhost:8080/tmwebrock/webServices/bobby/getAll

Status: 200 OK Time: 8 ms Size: 385 B

```
{
    "success": true,
    "isException": false,
    "isReturnSomething": true,
    "result": [
        {
            "rollNumber": 100,
            "name": "Mohit"
        },
        {
            "rollNumber": 200,
            "name": "Rajveer"
        },
        {
            "rollNumber": 300,
            "name": "Tom"
        },
        {
            "rollNumber": 400,
            "name": "Jack"
        }
    ]
}
```



Testing Module / New Request

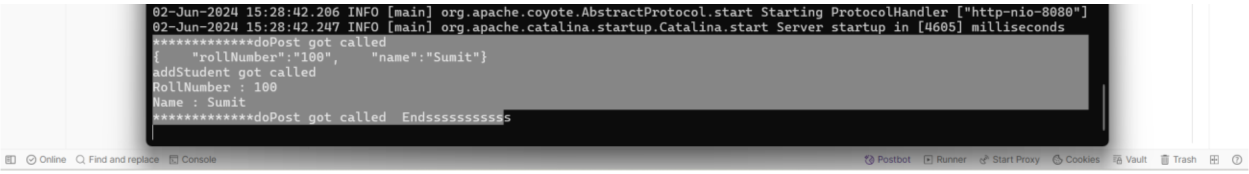
POST http://localhost:8080/tmwebrock/webServices/bobby/addStudent

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "rollNumber":"100",
    "name":"Sumit"
}
```

Status: 200 OK Time: 292 ms Size: 253 B

```
{
    "success": true,
    "isException": false,
    "isReturnSomething": true,
    "result": 1004
}
```



```
02-Jun-2024 15:28:42.206 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8080"]
02-Jun-2024 15:28:42.247 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in [4605] milliseconds
*************doPost got called
{    "rollNumber":"100",    "name":"Sumit"}
addStudent got called
RollNumber : 100
Name : Sumit
*************doPost got called  Endsssssssssss
```

If you want to use scopes in your application . You can simply use these classes.

@InjectApplicationScope

@InjectSessionScope

@InjectRequestScope

@InjectApplicationDirectory

You can simply write setAttribute method according to your scope, parameter specify according to your scope

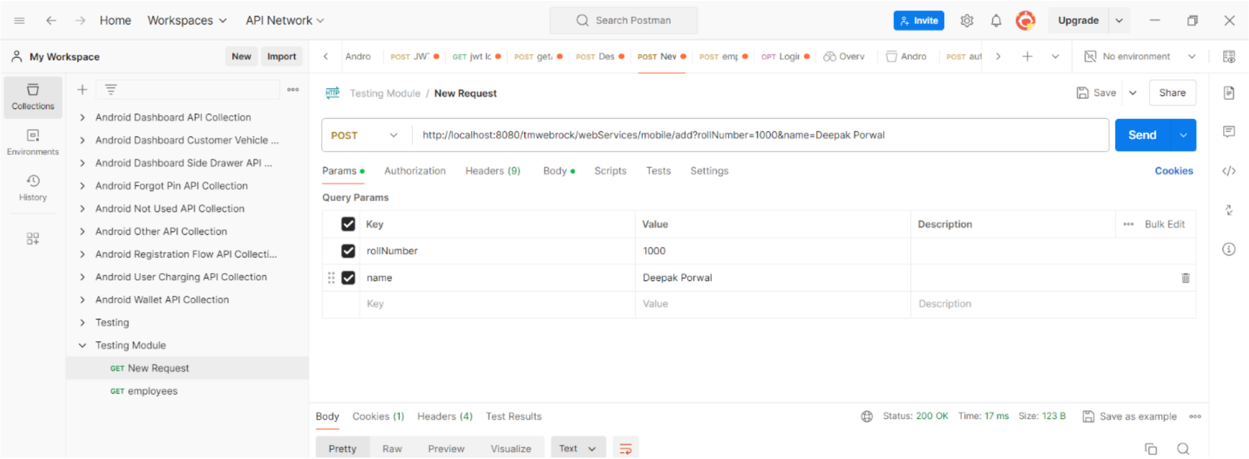
You can use scopes according to your requirement.

```
package bobby.com.thinking.machines;
import com.thinking.machines.webrock.annotations.*;
import com.thinking.machines.webrock.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
@Path("/mobile")
public class MobileService
{
// you can also use SessionScope,RequestScope same as
private ApplicationScope applicationScope;
public void setAttribute(ApplicationScope a)
{
this.applicationScope=a;
}
private ApplicationDirectory applicationDirectory;
public void setAttribute(ApplicationDirectory applicationDirectory)
{
this.applicationDirectory=applicationDirectory;
}
@Post
@Path("/add")
@InjectApplicationScope
@InjectApplicationDirectory
public void add(@RequestParameter("rollNumber") int
rollNumber,@RequestParameter("name") String n)
{

System.out.println("add got called Mobile Add (Module) got called");
System.out.println("RollNumber  : "+rollNumber);
System.out.println("Name : "+n);
ServletContext sc=this.applicationScope.getAttribute();
sc.setAttribute("rollNumber",rollNumber);
sc.setAttribute("name",n);
File file=this.applicationDirectory.getAttribute();
System.out.println("Path name : "+file.getAbsolutePath());
}

@Get
@Path("/get")
@InjectApplicationScope
public Object  get()
{
System.out.println("get got called from Mobile Service");
ServletContext sc=this.applicationScope.getAttribute();
int rollNumber=(Integer)sc.getAttribute("rollNumber");
String name=(String)sc.getAttribute("name");
Student student=new Student(rollNumber,name);
return student;
}

}
```

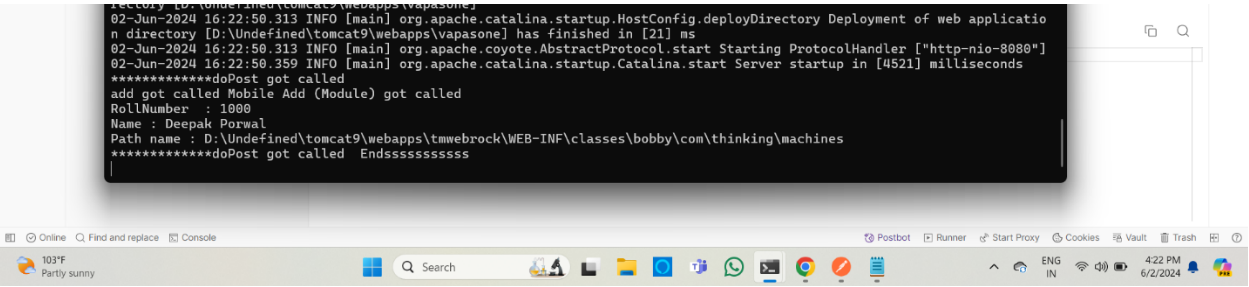

Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Collections
Environments
History
Android Dashboard API Collection
Android Dashboard Customer Vehicle ...
Android Dashboard Side Drawer API ...
Android Forgot Pin API Collection
Android Not Used API Collection
Android Other API Collection
Android Registration Flow API Collecti...
Android User Charging API Collection
Android Wallet API Collection
Testing
Testing Module
GET New Request
GET employees
No environment
Testing Module / New Request
Save
Share
Send
Params
Authorization
Headers (9)
Body
Scripts
Tests
Settings
Cookies
Bulk Edit
Body
Cookies (1)
Headers (4)
Test Results
Save as example
Pretty
Raw
Preview
Visualize
Text

POST http://localhost:8080/tmwebrock/webServices/mobile/add?rollNumber=1000&name=Deepak Porwal

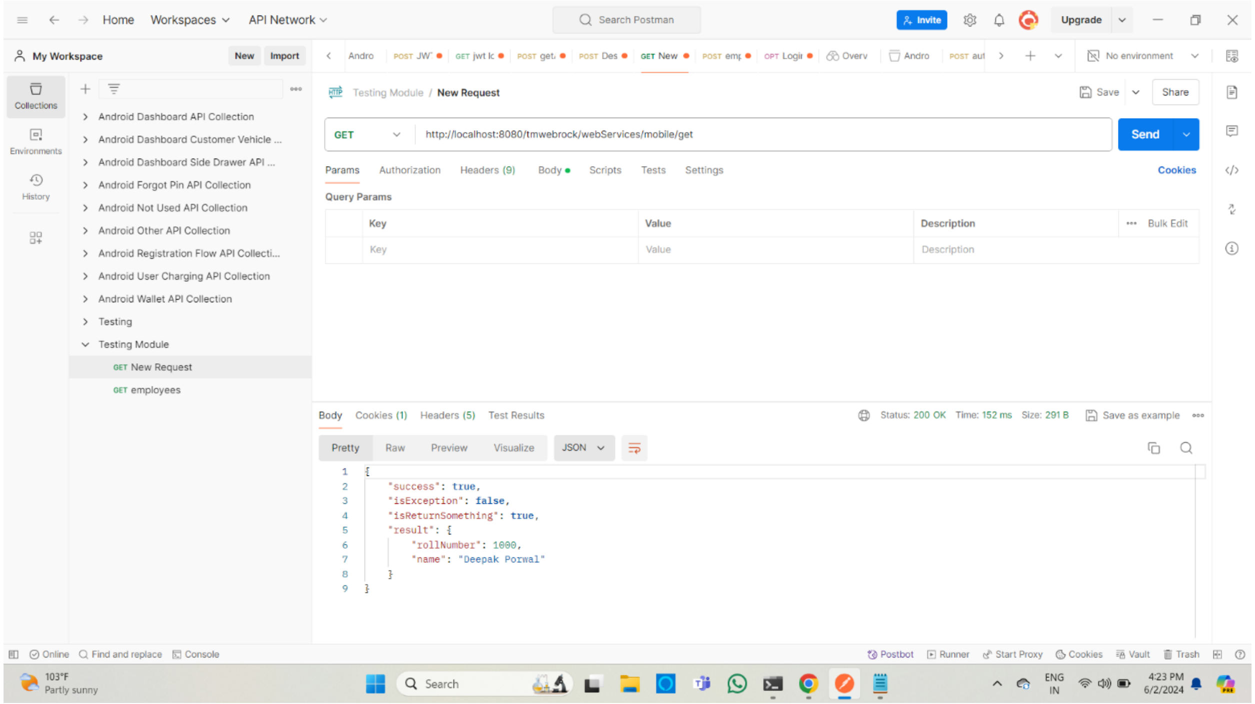
Query Params

☑ Key	Value	Description
☑ rollNumber	1000	
☑ name	Deepak Porwal	
Key	Value	Description

Status: 200 OK Time: 17 ms Size: 123 B



```
rectory [D:\Undefined\tomcat9\webapps\vapasone]
02-Jun-2024 16:22:50.313 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web applicatio
n directory [D:\Undefined\tomcat9\webapps\vapasone] has finished in [21] ms
02-Jun-2024 16:22:50.313 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8080"]
02-Jun-2024 16:22:50.359 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in [4521] milliseconds
*************doPost got called
add got called Mobile Add (Module) got called
RollNumber  : 1000
Name : Deepak Porwal
Path name : D:\Undefined\tomcat9\webapps\tmwebrock\WEB-INF\classes\bobby\com\thinking\machines
*************doPost got called  Endssssssssss
```



Testing Module / New Request

GET http://localhost:8080/tmwebrock/webServices/mobile/get

Status: 200 OK Time: 152 ms Size: 291 B

```
{
    "success": true,
    "isException": false,
    "isReturnSomething": true,
    "result": {
        "rollNumber": 1000,
        "name": "Deepak Porwal"
    }
}
```

You can also use Scopes in your method.

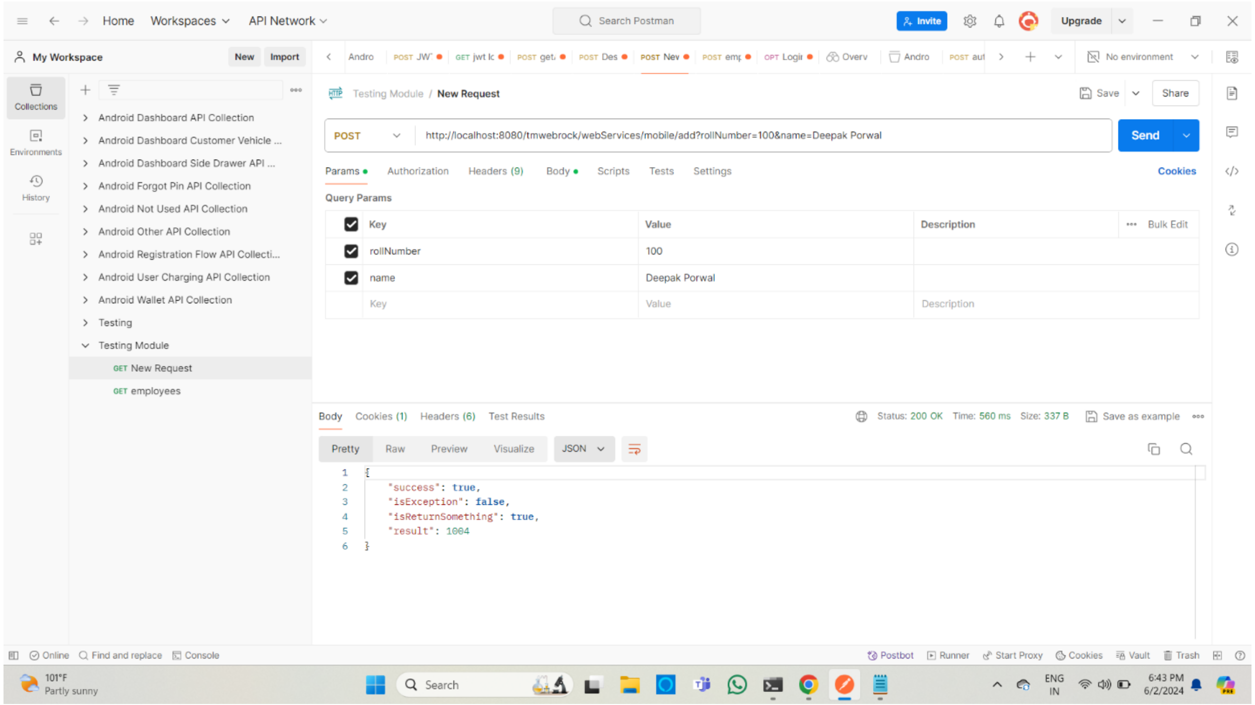
Instead use annotation.

you can simply write scopes in your method parameter according to your requirement.

```
package bobby.com.thinking.machines;
import com.thinking.machines.webrock.annotations.*;
import com.thinking.machines.webrock.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
@Path("/mobile")
public class MobileService
{
// you can also use SessionScope,RequestScope same as
@Post
@Path("/add")
public Object add(@RequestParameter("rollNumber") int
rollNumber,@RequestParameter("name") String n,ApplicationScope
applicationScope,ApplicationDirectory applicationDirectory)
{
System.out.println("add got called Mobile Add (Module) got called");
System.out.println("RollNumber  : "+rollNumber);
System.out.println("Name : "+n);
ServletContext sc=applicationScope.getAttribute();
sc.setAttribute("rollNumber",rollNumber);
sc.setAttribute("name",n);
File file=applicationDirectory.getAttribute();
System.out.println("Path name : "+file.getAbsolutePath());
return 1004; // Successfully added
}

@Get
@Path("/get")
public Object  get(ApplicationScope applicationScope)
{
System.out.println("get got called from Mobile Service");
ServletContext sc=applicationScope.getAttribute();
int rollNumber=(Integer)sc.getAttribute("rollNumber");
String name=(String)sc.getAttribute("name");
Student student=new Student(rollNumber,name);
return student;
}

}
```

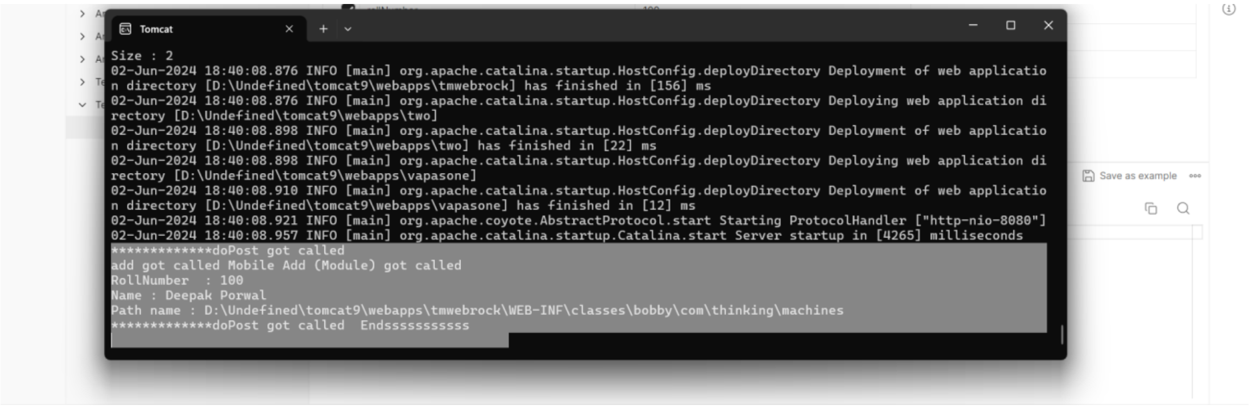


POST http://localhost:8080/tmwebrock/webServices/mobile/add?rollNumber=100&name=Deepak Porwal

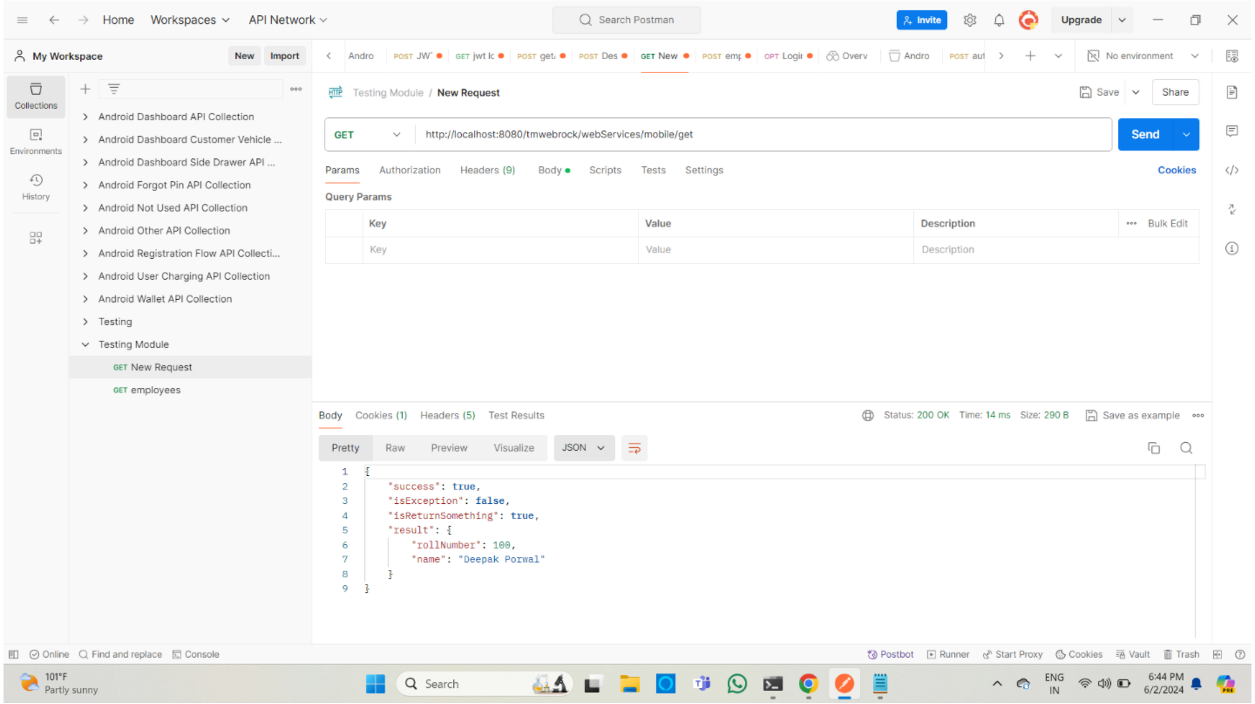
☑ Key	Value	Description
☑ rollNumber	100	
☑ name	Deepak Porwal	
Key	Value	Description

Status: 200 OK Time: 560 ms Size: 337 B

```
{
    "success": true,
    "isException": false,
    "isReturnSomething": true,
    "result": 1004
}
```



```
Size : 2
02-Jun-2024 18:40:08.876 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web applicatio
n directory [D:\Undefined\tomcat9\webapps\tmwebrock] has finished in [156] ms
02-Jun-2024 18:40:08.876 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application di
rectory [D:\Undefined\tomcat9\webapps\two]
02-Jun-2024 18:40:08.898 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web applicatio
n directory [D:\Undefined\tomcat9\webapps\two] has finished in [22] ms
02-Jun-2024 18:40:08.898 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application di
rectory [D:\Undefined\tomcat9\webapps\vapasone]
02-Jun-2024 18:40:08.910 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web applicatio
n directory [D:\Undefined\tomcat9\webapps\vapasone] has finished in [12] ms
02-Jun-2024 18:40:08.921 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8080"]
02-Jun-2024 18:40:08.957 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in [4265] milliseconds
*************doPost got called
add got called Mobile Add (Module) got called
RollNumber  : 100
Name : Deepak Porwal
Path name : D:\Undefined\tomcat9\webapps\tmwebrock\WEB-INF\classes\bobby\com\thinking\machines
*************doPost got called  Endsssssssssss
```



Testing Module / New Request

GET http://localhost:8080/tmwebrock/webServices/mobile/get

Status: 200 OK Time: 14 ms Size: 290 B

```
{
    "success": true,
    "isException": false,
    "isReturnSomething": true,
    "result": {
        "rollNumber": 100,
        "name": "Deepak Porwal"
    }
}
```

@Forward Annotation

Using this annotation user can forward request to another web service or to some client side technology like (html files) .The example below shows, How to use forward annotaton to forward request to other service, you can also forward to some JSP also, by giving JSP file name as value of forward annotation. Forward annotation can only be applied on Services(Method which has path annotation applied on it)

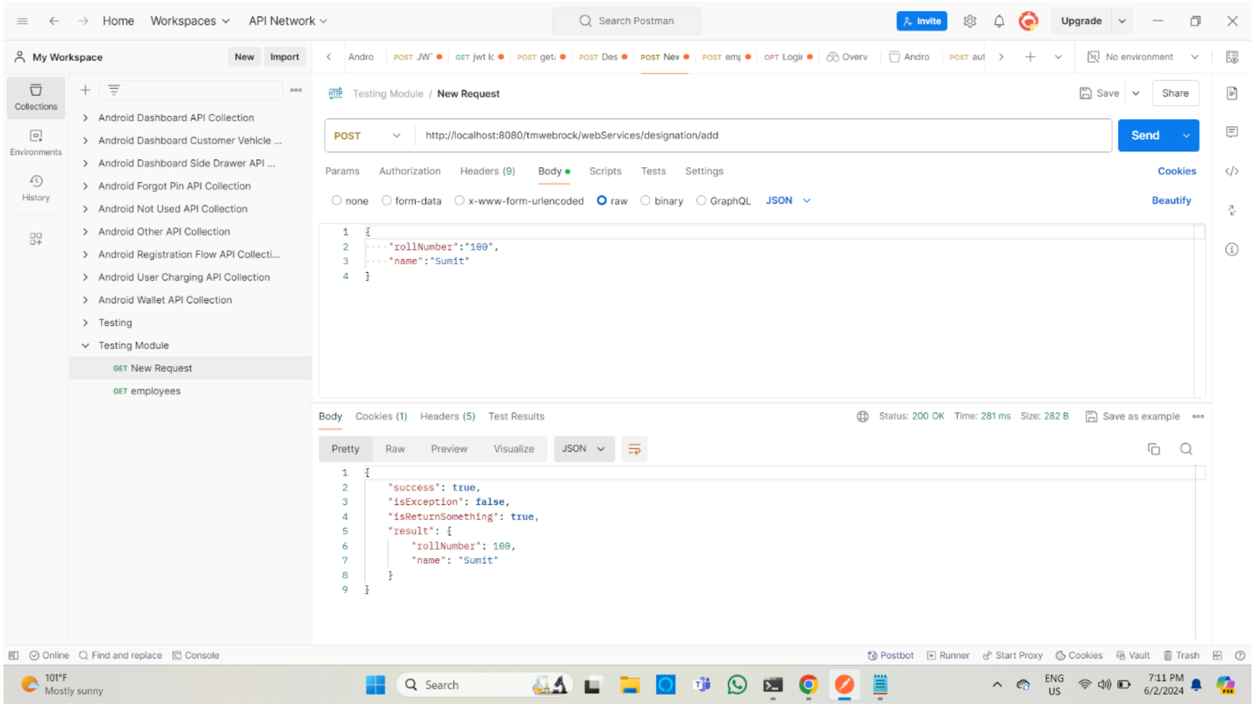
```
package bobby.com.thinking.machines;
import com.thinking.machines.webrock.annotations.*;
import com.thinking.machines.webrock.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

@Path("/designation")
public class DesignationService
{
@Post
@Path("/add")
@Forward("/designation/get")
public void add(Student student,SessionScope
sessionScope,ApplicationDirectory applicationDirectory)
{

System.out.println("add got called Designation Add (Module) got called");
System.out.println("RollNumber  : "+student.rollNumber);
System.out.println("Name : "+student.name);
HttpSession session=sessionScope.getAttribute();
session.setAttribute("student",student);
}

@Get
@Path("/get")
public Object  get(SessionScope sessionScope)
{
System.out.println("get got called from Designation Service");
HttpSession hs=sessionScope.getAttribute();
return hs.getAttribute("student");
}

}
```



Testing Module / New Request

POST http://localhost:8080/tmwebrock/webServices/designation/add

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "rollNumber":"100",
    "name":"Sumit"
}
```

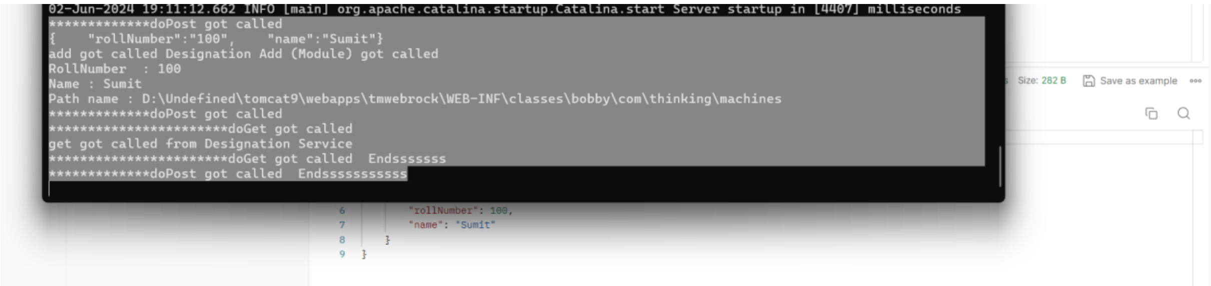
Body Cookies (1) Headers (5) Test Results

Status: 200 OK Time: 281 ms Size: 282 B

```
{
    "success": true,
    "isException": false,
    "isReturnSomething": true,
    "result": {
        "rollNumber": 100,
        "name": "Sumit"
    }
}
```

101°F
Mostly sunny

7:11 PM
6/2/2024



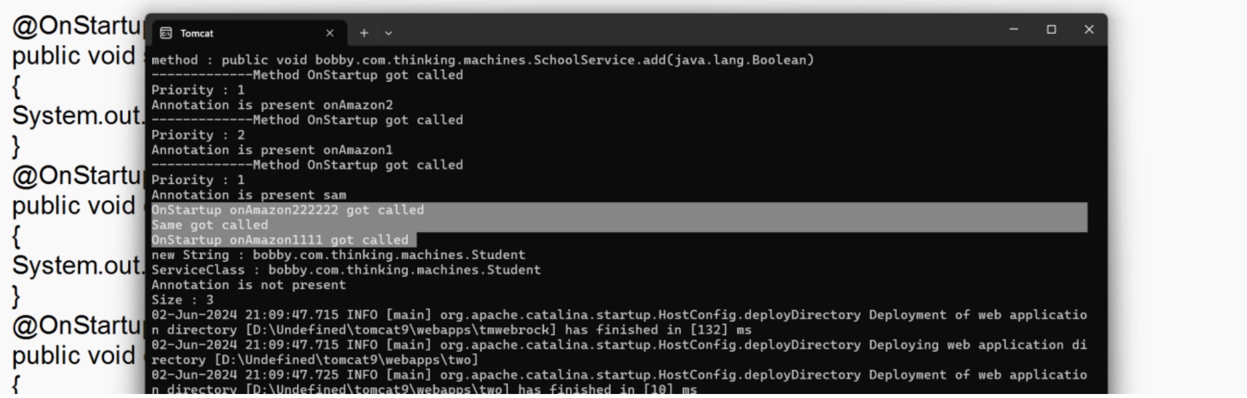
```
02-Jun-2024 19:11:12.662 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in [4407] milliseconds
*************doPost got called
{    "rollNumber":"100",    "name":"Sumit"}
add got called Designation Add (Module) got called
RollNumber  : 100
Name : Sumit
Path name : D:\Undefined\tomcat9\webapps\tmwebrock\WEB-INF\classes\bobby\com\thinking\machines
*************doPost got called
***********************doGet got called
get got called from Designation Service
***********************doGet got called  Endssssssss
*************doPost got called  Endssssssssssss
```

@OnStartup Annotation

This annotation can only be applied on method/service. User uses this annotation if user wants that one or more service got called when server get started then he/she must apply this annotation over those services and user have to mention priority of calling this services, lesser priority number will called first. By default is 1
Note: You are not supposed to apply Path annotation with OnStartup annotation. you cannot use RequestScope or SessionScope as a parameter. you can only use ApplicationScope as a parameter.

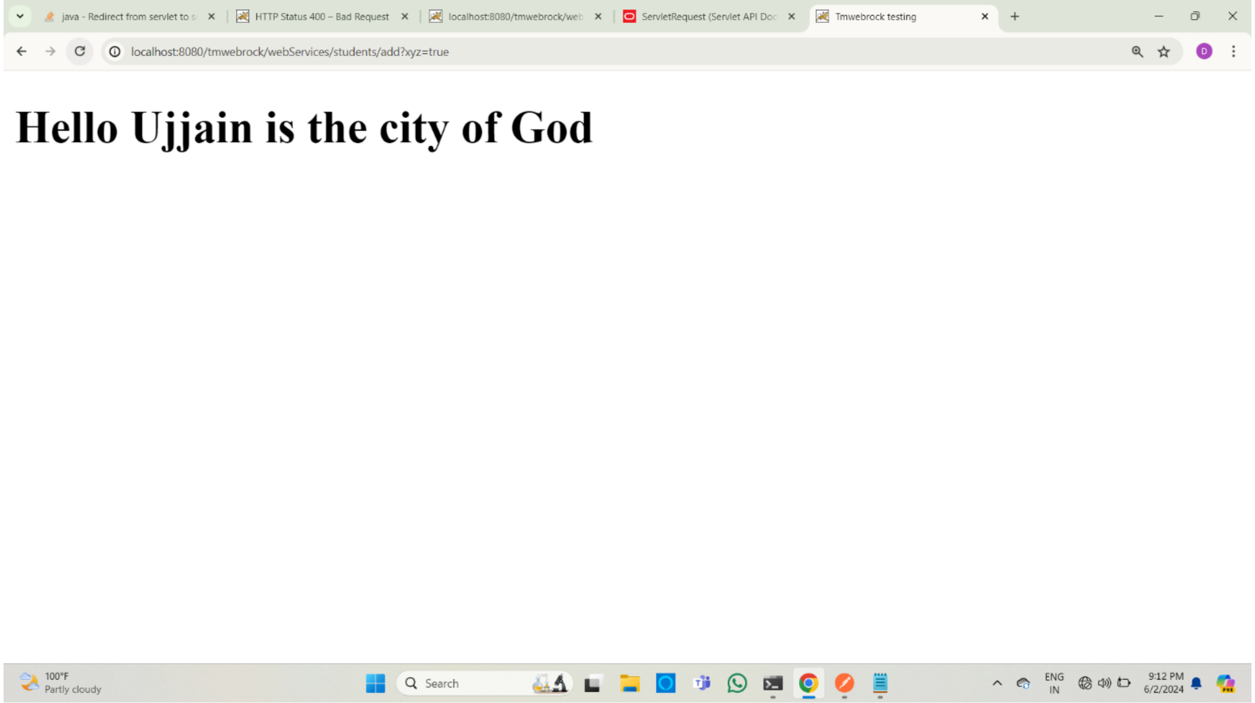
```
package bobby.com.thinking.machines;
import java.util.*;
import com.thinking.machines.webrock.annotations.*;
import com.thinking.machines.webrock.*;
import javax.servlet.*;
@Path("/students")
public class SchoolService
{
@Get
@Path("/add")
@Forward("/index.jsp")
// index.jsp is present in user-application-context/index.jsp
public void add(@RequestParameter("xyz") Boolean b)
{
System.out.println("Boolean value : "+b);
System.out.println("add got called School Detail got called");
}

@OnStartup(priority=1)
public void sam(ApplicationScope applicationScope)
{
System.out.println("Same got called");
ServletContext servletContext=applicationScope.getAttribute();
Student student=new Student(100,”Sumit”);
servletContext.setAttribute(“student”,student);
}
@OnStartup(priority=2)
public void onAmazon1()
{
System.out.println("OnStartup onAmazon1111 got called");
}
@OnStartup()
public void onAmazon2()
{
System.out.println("OnStartup onAmazon222222 got called");
}
}
```



```
@OnStartu
public void
{
System.out.
}
@OnStartu
public void
{
System.out.
}
@OnStartu
public void
{
```

```
method : public void bobby.com.thinking.machines.SchoolService.add(java.lang.Boolean)
-------------Method OnStartup got called
Priority : 1
Annotation is present onAmazon2
-------------Method OnStartup got called
Priority : 2
Annotation is present onAmazon1
-------------Method OnStartup got called
Priority : 1
Annotation is present sam
OnStartup onAmazon222222 got called
Same got called
OnStartup onAmazon1111 got called
new String : bobby.com.thinking.machines.Student
ServiceClass : bobby.com.thinking.machines.Student
Annotation is not present
Size : 3
02-Jun-2024 21:09:47.715 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web applicatio
n directory [D:\Undefined\tomcat9\webapps\tmwebrock] has finished in [132] ms
02-Jun-2024 21:09:47.715 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application di
rectory [D:\Undefined\tomcat9\webapps\two]
02-Jun-2024 21:09:47.725 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web applicatio
n directory [D:\Undefined\tomcat9\webapps\two] has finished in [10] ms
```



java - Redirect from servlet to s
HTTP Status 400 – Bad Request
localhost:8080/tmwebrock/web
ServletRequest (Servlet API Doc
Tmwebrock testing

localhost:8080/tmwebrock/webServices/students/add?xyz=true

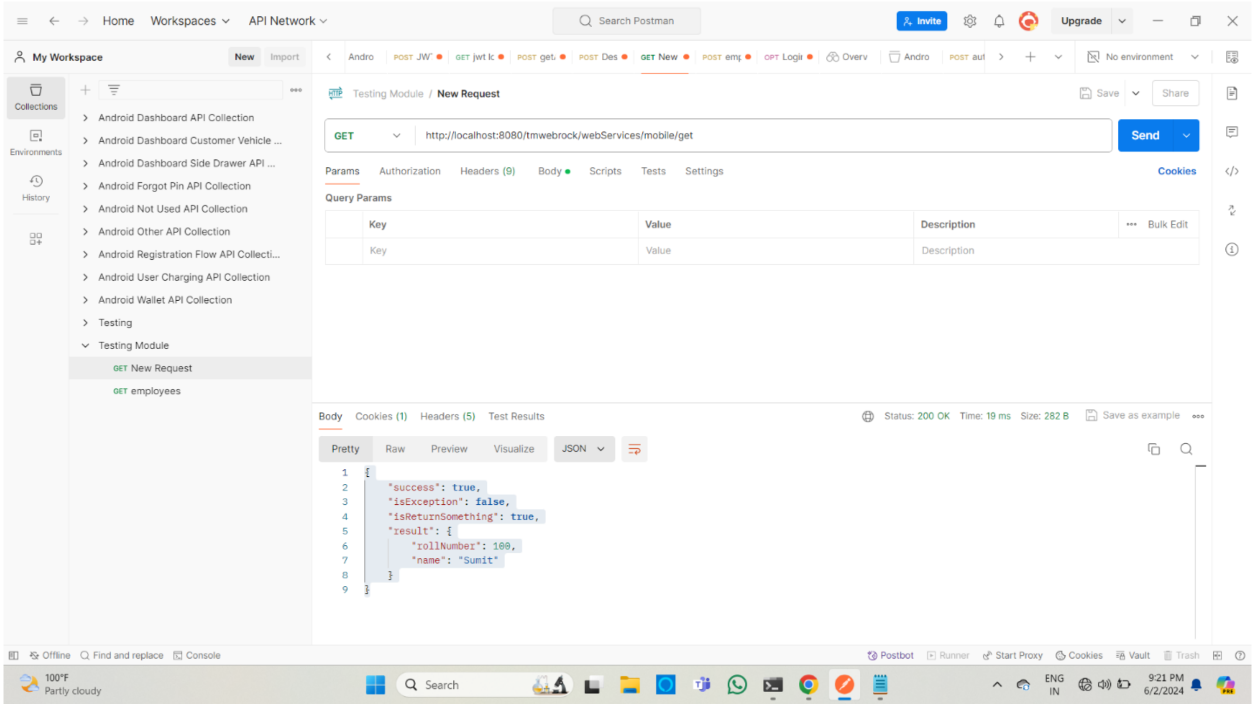
Hello Ujjain is the city of God

100°F
Partly cloudy
Search
ENG
IN
9:12 PM
6/2/2024

@AutoWird Annotation

AutoWired annotation can only be applied on properties of a service class. This annotation is used for binding the properties. If some data has been stored in some scope as key value pair. If the user wants that data, So, according to the key name the data will be extracted from the appropriate scope as value & that value will be assigned to the property which has AutoWired annotation applied on it. Setter method should be present for that property so that value can be setted. If the setter method is not found, then exception will be raised.
Firstly it will search on ApplicationScope then SessionScope then RequestScope if it is not found then the value is null

```
package bobby.com.thinking.machines;
import com.thinking.machines.webrock.annotations.*;
import com.thinking.machines.webrock.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
@Path("/mobile")
public class MobileService
{
@AutoWired(name="student")
private Student student;
public void setStudent(Student student)
{
this.student=student;
}
@Get
@Path("/get")
public Object  get()
{
System.out.println("get got called from Mobile Service");
return this.student;
}
}
```



GET http://localhost:8080/tmwebrock/webServices/mobile/get

Status: 200 OK Time: 19 ms Size: 282 B

```
{
    "success": true,
    "isException": false,
    "isReturnSomething": true,
    "result": {
        "rollNumber": 100,
        "name": "Sumit"
    }
}
```

@ResponseHeader Annotation

Http Headers

In the event that you need to include custom HTTP headers in your web service response, you can deploy the `ResponseHeaders` annotation. This annotation should be assigned a value representing an HTTP header provider class, which is required to implement the `CustomHttpHeaderProvider` interface.

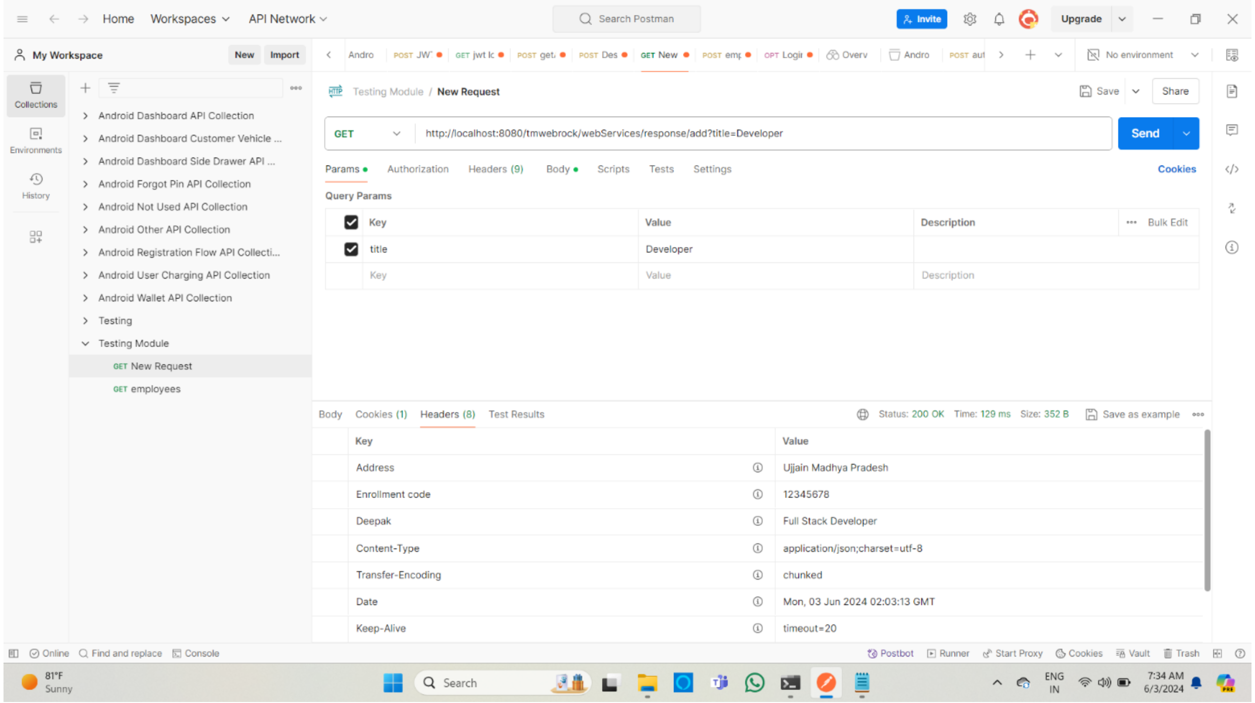
Apply @ResponseHeader on the service from which we want to attach the HTTP headers in response. The value of the provider
will be the class that implements the interface CustomHttpHeadersProvider.

```
package bobby.com.thinking.machines.security;
import java.util.*;
import com.thinking.machines.webrock.annotations.*;
import com.thinking.machines.webrock.*;
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class ResponseHeader implements CustomHttpHeadersProvider {
public HashMap<String,String> getCustomHttpHeaders(ServletContext
sc,HttpSession ht,HttpServletRequest hsr)
{
System.out.println("Header got called");
 HashMap<String, String> headers = new HashMap<>();
String tt=(String)ht.getAttribute("Deepak");
headers.put("Deepak",tt);
headers.put("Enrollment code","12345678");
headers.put("Address","Ujjain Madhya Pradesh");
        return headers;
    }
}
```

```
package bobby.com.thinking.machines;
import java.util.*;
import com.thinking.machines.webrock.annotations.*;
import com.thinking.machines.webrock.annotations.ResponseHeader;
import com.thinking.machines.webrock.*;
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
@Path("/response")
public class ResponseService
{
```

```
@Get
@Path("/add")
@ResponseHeader(provider=bobby.com.thinking.machines.ResponseHeader.class)
public Object add(@RequestParameter("title") String title,SessionScope sessionScope)
{
System.out.println("Title : "+title);
HttpSession ht=sessionScope.getAttribute();
ht.setAttribute("Deepak","Full Stack Developer");
System.out.println("add got called ResponseService Add (Module) got called");
return "Authenticate";
}

}
```



Testing Module / New Request

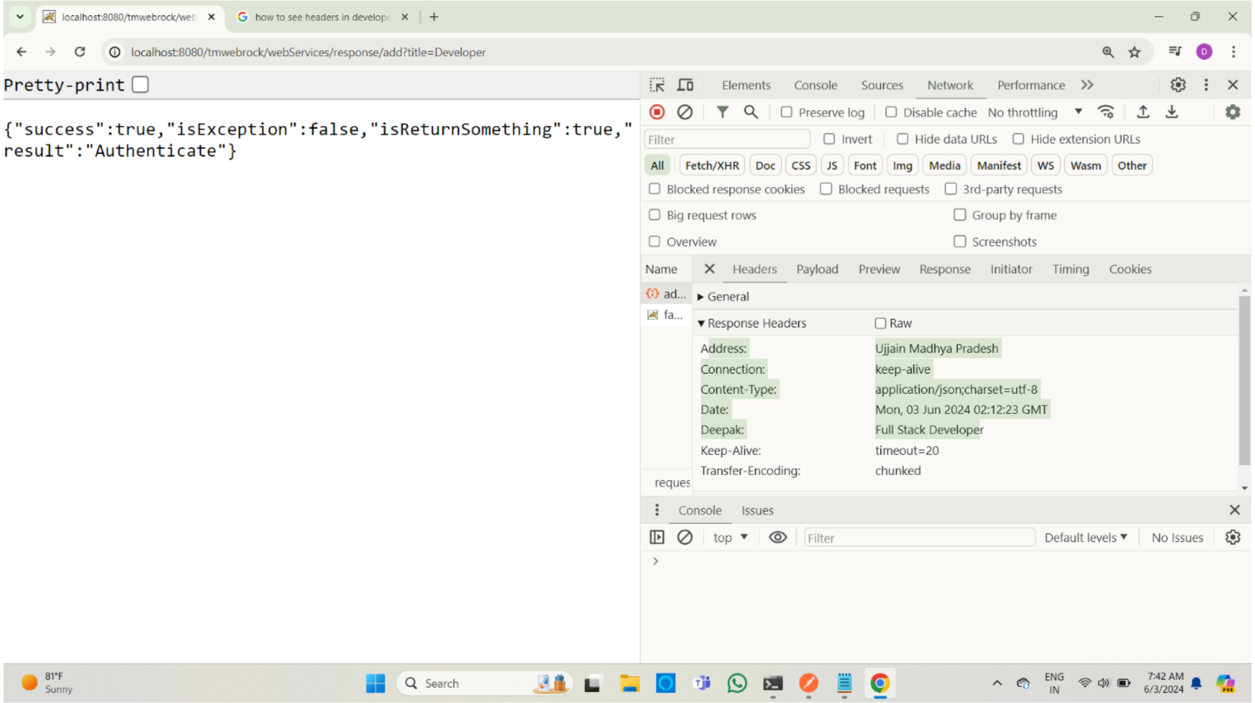
GET http://localhost:8080/tmwebrock/webServices/response/add?title=Developer

Query Params

	Key	Value	Description
☑	title	Developer	

Status: 200 OK Time: 129 ms Size: 352 B

Key	Value
Address	Ujjain Madhya Pradesh
Enrollment code	12345678
Deepak	Full Stack Developer
Content-Type	application/json;charset=utf-8
Transfer-Encoding	chunked
Date	Mon, 03 Jun 2024 02:03:13 GMT
Keep-Alive	timeout=20



localhost:8080/tmwebrock/webServices/response/add?title=Developer

Pretty-print □

```
{"success":true,"isException":false,"isReturnSomething":true,"
result":"Authenticate"}
```

▸ General

▾ Response Headers □ Raw

```
Address:            Ujjain Madhya Pradesh
Connection:         keep-alive
Content-Type:       application/json;charset=utf-8
Date:               Mon, 03 Jun 2024 02:12:23 GMT
Deepak:             Full Stack Developer
Keep-Alive:         timeout=20
Transfer-Encoding:  chunked
```

@Securd Annotation

Security Guard When it comes to web app security, having secure URLs is essential. We can control who can access the service by putting security guards on it. here is how we can do it,

Implement four methods in the guard

Method Name	Return Type	Parameter	Description
allow	boolean	1.String 2.ServletContext 3.HttpSession 4.HttpServletRequest	Through this method, we can decide whether we want to allow access to the protected method by this request or not. If this method returns **true** then the service will be allowed to access otherwise not.
divertTo	TMForward	1.String 2.ServletContext 3.HttpSession 4.HttpServletRequest	This method returns TMForward class object. We can provide the forwarding url in the parameterised constructor of TMForward.
redirectTo	TMRedirect	1.String 2.ServletContext 3.HttpSession 4.HttpServletRequest	This method returns TMRedirect class object. We can provide the redirecting url in the parameterised constructor of TMRedirect.
getAction Type	String		This method provides action type. If guard want to redirect the method then return “**redirect**” or if guard want to forward the method then return “**forward**”.

The following example shows how can we make secured services.

In this example, we have created both guards to redirect and forward the request but in a real scenario, you can either redirect the request or forward the request depending on the situation. So use guards accordingly. To make this example we have created four files 1.

..\webapps\tmwebrock \ index.jsp This is the jsp page which will be displayed when an unknown user wants to access the secured service.

```
<!DOCTYPE HTML>
<html lang='en'>
<head>
<meta charset='utf-8'>
<title>Tmwebrock testing</title>
</head>
<body>
<h1>Unauthorized Access Please try again later</h1>
</body>
</html>
```

The **allow** method is checking the existence of admin in the HTTP session. It returns true if the admin is present in the session and returns false if the session is not present in the HTTP session. The **getActionType** method return “**redirect”** because this guard redirects the request. The **redirectTo** method returns TMRedirect object, In this, we have put the request redirecting URL as **/tmwebrock /index.jsp** In the **divertTo ,** do nothing just return **TMForward(“”)**

```
package bobby.com.thinking.machines.security;
import java.util.*;
import com.thinking.machines.webrock.annotations.*;
import com.thinking.machines.webrock.*;
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class MobileGuard implements SecurityGuard{

public boolean allow(String str, ServletContext servletContext, HttpSession
httpSession, HttpServletRequest httpServletRequest)
{
try
{
System.out.println(str);
httpSession.setAttribute("name","Deepak Porwal");
return false;
}catch(Exception e){
return false;
}
}
    public TMForward divertTo(String str, ServletContext servletContext, 
HttpSession httpSession, HttpServletRequest httpServletRequest) {
System.out.println("Divert to got called");
       return new TMForward("");
    }

    public String getActionType() {
        return "redirect";
             }

    public TMRedirect redirectTo(String str, ServletContext servletContext, 
HttpSession httpSession, HttpServletRequest httpServletRequest) {
        return new TMRedirect("/index.jsp");
    }
}
```

The **allow** method is checking the existence of admin in the HTTP session. It returns true if the admin is present in the session and returns false if the session is not present in the HTTP session. The **getActionType** method returns **"forward"** because this guard forwards the request. The method returns **TMForward** object, In this, we have put the request forwarding URL as **/mobileService/get** In the **redirectTo,** do nothing just return **TMRedirect("")**

```
package bobby.com.thinking.machines.security;
import java.util.*;
import com.thinking.machines.webrock.annotations.*;
import com.thinking.machines.webrock.*;
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class MobileGuard implements SecurityGuard{

public boolean allow(String str, ServletContext servletContext, HttpSession
httpSession, HttpServletRequest httpServletRequest)
{
try
{
System.out.println(str);
httpSession.setAttribute("name","Deepak Porwal");

return false;
}catch(Exception e){
return false;
}
}
   public TMForward divertTo(String str, ServletContext servletContext,
HttpSession httpSession, HttpServletRequest httpServletRequest) {
System.out.println("Divert to got called");
      return new TMForward("/mobileService/get");
   }

   public String getActionType() {
      return "forward";
          }

   public TMRedirect redirectTo(String str, ServletContext servletContext,
HttpSession httpSession, HttpServletRequest httpServletRequest) {
      return new TMRedirect("");
   }
}
```

We have created the **add** service and it should only be allowed if the admin is present in the HTTP session. To do that we are using @Secured annotation and providing the annotation value according to the situation.

To redirect the request we can use -
We use redirect

To forward the request we can use-
We use forward

```
package bobby.com.thinking.machines;
import java.util.*;
import com.thinking.machines.webrock.annotations.*;
import com.thinking.machines.webrock.*;
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;


class Student
{// i can also go for private property
public int rollNumber;
public String name;
public Student(int rollNumber,String name)
{
this.rollNumber=rollNumber;
this.name=name;
}
//Setter Getter
}

@Path("/mobileService")
public class MobileGuardService
{
@Get
@Path("/add")
@Secured(guard=bobby.com.thinking.machines.security.MobileGuard.class)
public Object add(@RequestParameter("name") String name,SessionScope
sessionScope,RequestScope requestScope)
{
HttpSession ht=sessionScope.getAttribute();
ht.setAttribute("Deepak","Volttic Ev Charging Solution");
ht.setAttribute("Mahesh Parmar","Charge Point");
System.out.println("add got called MobileGuard Add (Module) got called");
Student s=new Student(100,"Deepak Porwal");
return s;
}
@Post
@Path("/get")
public Object  get(RequestScope requestScope,SessionScope sessionScope)
{
HttpSession session=sessionScope.getAttribute();
System.out.println(session.getAttribute("name"));
System.out.println("get got called from (MobileGuard) ");
return session.getAttribute("name")+"Forwarding";
}
}
```

Annotation Summary

Annotation	Value	Description
@Path	String	This annotation is used by the framework to find the web service and web service class. The value of this annotation should be unique.
@Get	--	This annotation is used to specify the method type as GET for the web service.
@Path	--	This annotation is used to specify the method type as POST of the web service.
@InjectApplicationScope	--	This annotation is used to inject ServletContext/Application object for the web service
@InjectSessionScope	--	This annotation is used to inject Session Object for the web service.
@InjectRequestScope	--	This annotation is used to inject Request object for the web service.
@InjectApplicationDirectory	--	This annotation is used to inject the directories for the web service.
@OnStartup	int	This annotation is used to specify the startup service, means the service which should be run when server is starts. The value of this annotation is determines the sequence of the webservice where where one means first and 2 means second and so on.
@AutoWired	String	The Autowired annotation can only be applied to properties of a service class. It binds properties to values stored in a scoped key-value store. When a user requests data, the value corresponding to the key is retrieved and assigned to the annotated property.
@Forward	String	The Forward annotation, used on Path-annotated service methods, allows forwarding requests to another web service or client-side technology (e.g., HTML or JSP files). Specify the JSP file name as the value of the Forward annotation.

@RequestParameter	String	The RequestParameter annotation can only be applied to parameters. It retrieves data from the web request's bag based on the specified entity. This annotation should be used on all primitive types.
@ResponseHeader	Class	you can effectively define and attach your own set of HTTP headers to the response generated by your web service.Header is class that implements CustomHttpHeadersProvider
@Secured	Class	This annotation is used to make secured web service. Basically, we provide a guard to check authenticate the request. Guard is a class that implements SecurityGuardInterface.

------------------------------End-----------------------------