Dinesh Pamu

B2- 30

ASSIGNMENT NO: 7

Title:

Simple web service and a distributed application to consume the web service.

Problem Statement:

Create a simple web service and write any distributed application to consume the web service.

Software and tools:

Programming language: Java

Framework: SpringBoot

IDE: Spring ToolSuite4

Networking: Java Sockets

Theory:

- A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message, then waits for a corresponding XML response. As all communication is in XML, web services are not tied to any one operating system or programming language—Java can talk with Perl; Windows applications can talk with Unix applications.
- Web services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains. These applications can be local, distributed, or web-based. Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML.
- Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents.
- A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.

To summarize, a complete web service is, therefore, any service that −

- Is available over the Internet or private (intranet) networks
- Uses a standardized XML messaging system
- Is not tied to any one operating system or programming language
- Is self-describing via a common XML grammar
- Is discoverable via a simple find mechanism

Components of Web Services

The basic web services platform is XML + HTTP. All the standard web services work using the following components −

- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language)

All these components have been discussed in the Web Services Architecture chapter.

How Does a Web Service Work?

A web service enables communication among various applications by using open standards such as HTML, XML, WSDL, and SOAP. A web service takes the help of −

- XML to tag the data
- SOAP to transfer a message
- WSDL to describe the availability of service.

You can build a Java-based web service on Solaris that is accessible from your Visual Basic program that runs on Windows.

You can also use C# to build new web services on Windows that can be invoked from your web application that is based on JavaServer Pages (JSP) and runs on Linux.

Example

Consider a simple account-management and order processing system. The accounting personnel use a client application built with Visual Basic or JSP to create new accounts and enter new customer orders.

The processing logic for this system is written in Java and resides on a Solaris machine, which also interacts with a database to store information.

The steps to perform this operation are as follows −

- The client program bundles the account registration information into a SOAP message.
- This SOAP message is sent to the web service as the body of an HTTP POST request.
- The web service unpacks the SOAP request and converts it into a command that the application can understand.
- The application processes the information as required and responds with a new unique account number for that customer.
- Next, the web service packages the response into another SOAP message, which it sends back to the client program in response to its HTTP request.
- The client program unpacks the SOAP message to obtain the results of the account registration process.

Conclusion:

In this practical, we have learnt how to create and use web services. However, a web service also include components such as WSDL, UDDI, and SOAP that contribute to make it active. Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents.

Code & Output:

Server side code:

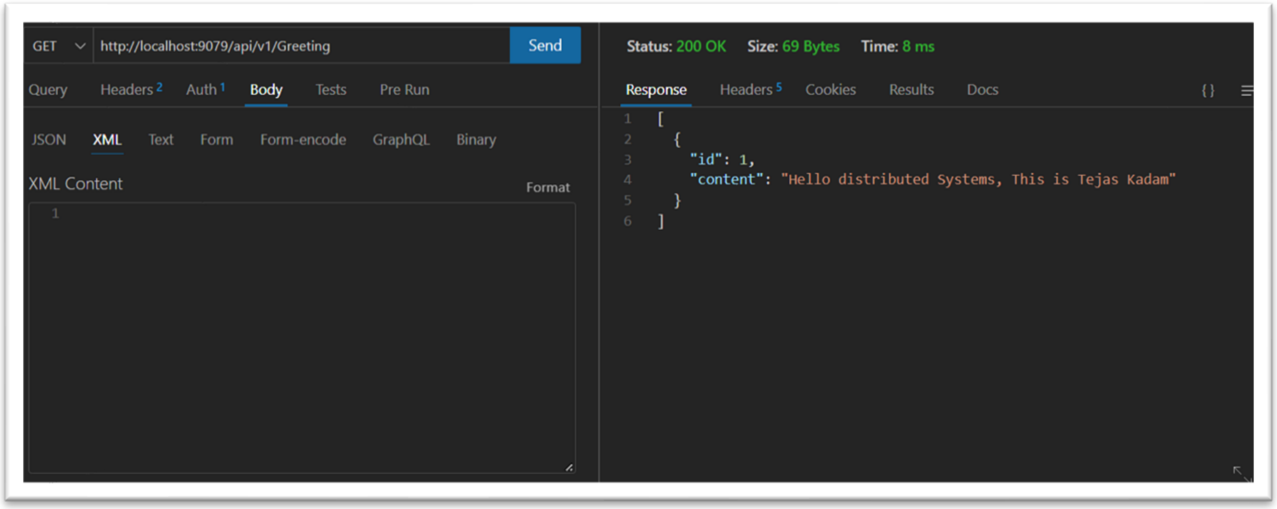
Pojo class:

```
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name = "testing")
public class testing {
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
        @Column(name = "content")
    private String content;

        public testing() {
                // TODO Auto-generated constructor stub
        }

    public testing(long id, String content) {
        this.id = id;
        this.content = content;
    }

    public long getId() {
        return id;
    }

    public String getContent() {
        return content;
    }

}
```

Controller class:

```
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com.example.demo.exception.ResourceNotFoundException;
import com.example.demo.model.Employee;
import com.example.demo.model.testing;
import com.example.demo.repositery.EmployeeRepository;
import com.example.demo.repositery.TestingRepo;

@CrossOrigin(origins = "http://localhost:4200/")
@RestController
@RequestMapping("/api/v1/")
public class EmployeeController {



        @Autowired
        TestingRepo tt;

        @GetMapping("/Greeting")
        public List<testing> GetGreeting(){

                return tt.findAll();
        }
```

<u>Server side API fetch Output:</u>



GET http://localhost:9079/api/v1/Greeting Send

Status: 200 OK Size: 69 Bytes Time: 8 ms

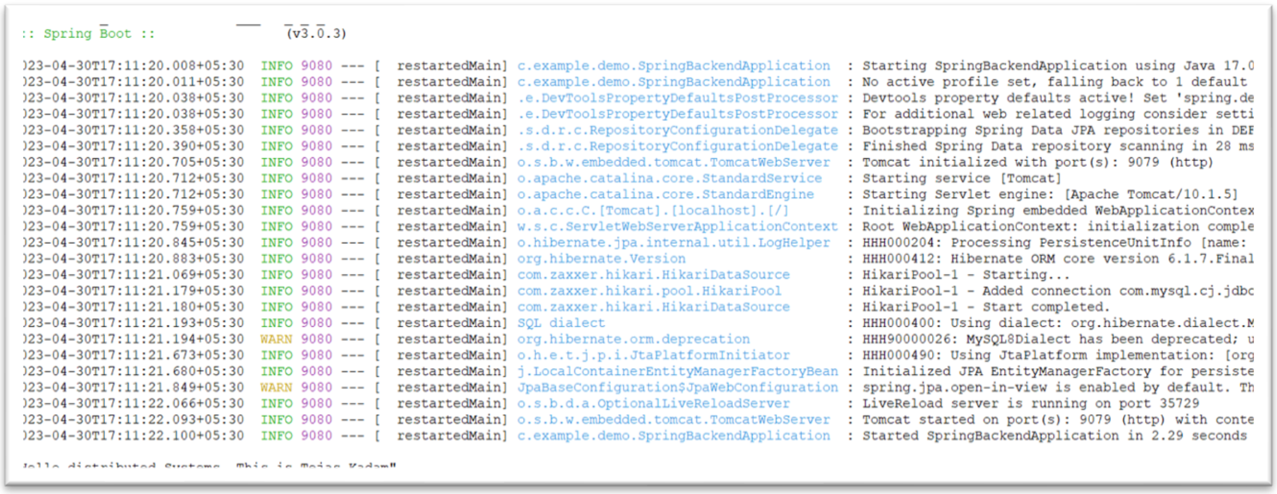
```
[
  {
    "id": 1,
    "content": "Hello distributed Systems, This is Tejas Kadam"
  }
]
```

Client-side code:

```
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class client extends SpringBootServletInitializer {

  @Bean
  public Jaxb2Marshaller marshaller() {
    Jaxb2Marshaller marshaller = new Jaxb2Marshaller();
    marshaller.setContextPath("com.example.consumingwebservice.wsdl");
```

```
        return marshaller;
    }

    @Bean
    public GreetingClient greetingClient(Jaxb2Marshaller marshaller) {
        GreetingClient client = new GreetingClient();
        client.setDefaultUri("http://localhost:8080/ws");
        client.setMarshaller(marshaller);
        client.setUnmarshaller(marshaller);
        return client;
    }
    public static void main(String[] args) {
        SpringApplication.run(GreetingClient.class, args);
    }
    @Autowired
    private GreetingClient greetingClient;

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
        return application.sources(GreetingClient.class);
    }

    @EventListener(ApplicationReadyEvent.class)
    public void doSomethingAfterStartup() {
        GetGreetingRequest request = new GetGreetingRequest();
        request.setName("John");
        GetGreetingResponse response = greetingClient.getGreeting(request);
        System.out.println(response.getGreeting().getContent());
    }
}
```

Client-side output:



```
:: Spring Boot ::                (v3.0.3)

)23-04-30T17:11:20.008+05:30  INFO 9080 --- [  restartedMain] c.example.demo.SpringBackendApplication  : Starting SpringBackendApplication using Java 17.0
)23-04-30T17:11:20.011+05:30  INFO 9080 --- [  restartedMain] c.example.demo.SpringBackendApplication  : No active profile set, falling back to 1 default
)23-04-30T17:11:20.038+05:30  INFO 9080 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.de
)23-04-30T17:11:20.038+05:30  INFO 9080 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setti
)23-04-30T17:11:20.358+05:30  INFO 9080 --- [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEF
)23-04-30T17:11:20.390+05:30  INFO 9080 --- [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 28 ms
)23-04-30T17:11:20.705+05:30  INFO 9080 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 9079 (http)
)23-04-30T17:11:20.712+05:30  INFO 9080 --- [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
)23-04-30T17:11:20.712+05:30  INFO 9080 --- [  restartedMain] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.5]
)23-04-30T17:11:20.759+05:30  INFO 9080 --- [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContex
)23-04-30T17:11:20.759+05:30  INFO 9080 --- [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization comple
)23-04-30T17:11:20.845+05:30  INFO 9080 --- [  restartedMain] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [name:
)23-04-30T17:11:20.883+05:30  INFO 9080 --- [  restartedMain] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 6.1.7.Final
)23-04-30T17:11:21.069+05:30  INFO 9080 --- [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
)23-04-30T17:11:21.179+05:30  INFO 9080 --- [  restartedMain] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mysql.cj.jdbc
)23-04-30T17:11:21.180+05:30  INFO 9080 --- [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
)23-04-30T17:11:21.193+05:30  INFO 9080 --- [  restartedMain] SQL dialect                              : HHH000400: Using dialect: org.hibernate.dialect.M
)23-04-30T17:11:21.194+05:30  WARN 9080 --- [  restartedMain] org.hibernate.orm.deprecation            : HHH90000026: MySQL8Dialect has been deprecated; u
)23-04-30T17:11:21.673+05:30  INFO 9080 --- [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform implementation: [org
)23-04-30T17:11:21.680+05:30  INFO 9080 --- [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persiste
)23-04-30T17:11:21.849+05:30  WARN 9080 --- [  restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Th
)23-04-30T17:11:22.066+05:30  INFO 9080 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
)23-04-30T17:11:22.093+05:30  INFO 9080 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 9079 (http) with conte
)23-04-30T17:11:22.100+05:30  INFO 9080 --- [  restartedMain] c.example.demo.SpringBackendApplication  : Started SpringBackendApplication in 2.29 seconds

Hello distributed Systems. This is Tejas Kadam"
```