

10) Aim: Basic RESTful Web Service

Description:

In this experiment, we will create a **Basic RESTful Web Service** using **Spring Boot**. The application demonstrates how to perform CRUD (Create, Read, Update, Delete) operations on an in-memory list of products through RESTful APIs. It allows adding, viewing, updating, and deleting products using standard HTTP methods like **GET**, **POST**, **PUT**, and **DELETE**.

- **DemoApplication.java** – Main class to run the Spring Boot application.
- **Product.java** – Model class representing a product with id, name, and price.
- **ProductController.java** – REST controller handling all product-related operations.
- **application.properties** – Configuration file for application name and server port.
- **pom.xml** – Maven configuration file for project dependencies.

Program:

ProductController.java

```
package com.example;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import org.springframework.web.bind.annotation.DeleteMapping;
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;

@RestController
@RequestMapping("/products")

public class ProductController {
    private List<Product> products = new ArrayList<>(
        Arrays.asList(
            new Product(1,"Laptop",53000),
            new Product(2,"mobile",20000)
```

```

        )
    };

    @GetMapping
    public List<Product> getAllProducts() {
        return products;
    }

    @GetMapping("/{id}")
    public Product getProductById(@PathVariable int id) {
        return products.stream()
            .filter(p -> p.getId() == id)
            .findFirst()
            .orElse(null);
    }

    @PostMapping
    public Product addProduct(@RequestBody Product product) {
        products.add(product);
        return product;
    }

    @PutMapping("/{id}")
    public Product updateProduct(@PathVariable int id ,@RequestBody Product
    updatedProduct ) {
        if(p.getId() == id) {
            p.setName(updatedProduct.getName());
            p.setPrice(updatedProduct.getPrice());
            return p;
        }
        return null;
    }

    @DeleteMapping("/{id}")
    public String deleteProduct(@PathVariable int id) {
        products.removeIf(p -> p.getId() ==id);
        return "Product with ID" +id+"delected!";
    }
}

```

application.properties

```

spring.application.name=Demo
server.port=6754

```

Demo.java

```
package com.example;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

}
```

Product.java

```
package com.example;

public class Product {
    private int id;
    private String name;
    private int Price;

    public Product() {
        super();
    }

    public Product(int id, String name, int price) {
        super();
        this.id = id;
        this.name = name;
        Price = price;
    }

    public int getId() {return id;}
    public void setId(int id) {this.id = id;}
    public String getName() {return name;}
    public void setName(String name) {this.name = name;}
    public int getPrice() {return Price;}
    public void setPrice(int price) {Price = price;}

}
```

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.5.6</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.example</groupId>
  <artifactId>DemoApplication</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Demo</name>
  <properties>
    <java.version>17</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
</project>
```

Output:

```

C:\Program Files\Java\jdk-17\bin\javaw.exe [02-Nov-2025, 8:56:53 pm elapsed: 0:00:57] [pid: 2352]

:: Spring Boot ::
(v3.5.6)

2025-11-02T20:56:59.428+05:30 INFO 2352 --- [Demo] [
main] c.o.e.example.DemoApplication 2352 : Starting DemoApplication
2025-11-02T20:56:59.444+05:30 INFO 2352 --- [Demo] [
main] c.o.e.example.DemoApplication 2352 : No active profile set, fall
2025-11-02T20:57:01.583+05:30 INFO 2352 --- [Demo] [
main] o.s.b.w.embedded.tomcat.TomcatWebServer 2352 : Tomcat initialized with
2025-11-02T20:57:01.618+05:30 INFO 2352 --- [Demo] [
main] o.apache.catalina.core.StandardService 2352 : Starting service [Tomcat]
2025-11-02T20:57:01.620+05:30 INFO 2352 --- [Demo] [
main] o.apache.catalina.core.StandardEngine 2352 : Starting Servlet engine: [
2025-11-02T20:57:01.740+05:30 INFO 2352 --- [Demo] [
main] o.a.c.c.C.[Tomcat].[localhost].[/] 2352 : Initializing Spring embedd
2025-11-02T20:57:01.741+05:30 INFO 2352 --- [Demo] [
main] w.s.c.ServletWebServerApplicationContext 2352 : Root WebApplicationConte
2025-11-02T20:57:02.849+05:30 INFO 2352 --- [Demo] [
main] o.s.b.w.embedded.tomcat.TomcatWebServer 2352 : Tomcat started on port 444
2025-11-02T20:57:02.869+05:30 INFO 2352 --- [Demo] [
main] c.o.e.example.DemoApplication 2352 : Started DemoApplication in

```

The image displays three sequential screenshots of a REST client interface, likely Postman, used for testing REST API endpoints. Each screenshot shows a different HTTP method and the resulting JSON response.

First Screenshot: The interface shows a **POST** request to `http://localhost:6754/products`. The **Body** tab is selected, displaying a JSON array with two product objects: `[{"id":1,"name":"Laptop","price":53000}, {"id":2,"name":"mobile","price":20000}]`. The **Send** button is visible on the right.

Second Screenshot: The interface shows a **PUT** request to `http://localhost:6754/products/2`. The **Body** tab is selected, displaying a JSON object: `{"id": 3, "name": "Headphones", "price": 3000}`. The **Send** button is visible on the right.

Third Screenshot: The interface shows a **PUT** request to `http://localhost:6754/products`. The **Body** tab is selected, displaying a JSON array with three product objects: `[{"id":1,"name":"Laptop","price":53000}, {"id":2,"name":"mobile","price":20000}, {"id":3,"name":"Headphones","price":3000}]`. The **Send** button is visible on the right.

11) Aim: Pagination and Sorting in RESTful Web Service

Description:

In this experiment, we will create a Spring Boot RESTful Web Service that demonstrates pagination and sorting features. The application manages an in-memory list of products and allows users to retrieve data in pages while sorting by id, name, or price in ascending or descending order.

- **PagingAndSortingApplication.java** – Main class to start the application.
- **Product.java** – Model class representing a product with id, name, and price.
- **ProductController.java** – REST controller handling requests for pagination and sorting.
- **application.properties** – Configuration file for application name and port.
- **pom.xml** – Maven configuration file for dependencies.

Program:

ProductController.java

```
package com.example;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/products")
public class ProductController {
    private List<Product> products = new ArrayList<>(
        Arrays.asList(
            new Product(1,"Laptop",53000),
            new Product(2,"Mobile",20000),
            new Product(3,"Tablet",15000),
            new Product(4,"Headphones",3000),
            new Product(5,"SmartWatch",8000),
            new Product(6,"Camera",25000),
```

```

        new Product(7,"Monitor",12000)
    )
);
@GetMapping
public List<Product> getProducts(
    @RequestParam(defaultValue = "0") int page,
    @RequestParam(defaultValue = "3") int size,
    @RequestParam(defaultValue = "id") String sortBy,
    @RequestParam(defaultValue = "asc") String order
){
    Comparator<Product> comparator = switch (sortBy) {
        case "name" -> Comparator.comparing(Product::getName);
        case "price" -> Comparator.comparing(Product::getPrice);
        default -> Comparator.comparing(Product::getId);
    };

    if (order.equalsIgnoreCase("desc")) {
        comparator = comparator.reversed();
    }

    List<Product> sortedList = new ArrayList<>(products);
    sortedList.sort(comparator);
    int start = page * size;
    int end = Math.min(start + size, sortedList.size());

    if (start >= sortedList.size()) return Collections.emptyList();
    return sortedList.subList(start, end);
}
}

```

PagingAndSortingApplication.java

```

package com.example;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class PagingAndSortingApplication {
    public static void main(String[] args) {
        SpringApplication.run(PagingAndSortingApplication.class, args);
    }
}

```

Product.java

```
package com.example;

public class Product {
    private int id;
    private String name;
    private double price;

    public Product() {
        super();
        // TODO Auto-generated constructor stub
    }
    public Product(int id, String name, double price) {
        super();
        this.id = id;
        this.name = name;
        this.price = price;
    }

    public int getId() {return id;}
    public void setId(int id) {this.id = id;}
    public String getName() {return name;}
    public void setName(String name) {this.name = name;}
    public double getPrice() {return price;}
    public void setPrice(double price) {this.price = price;}
}
```

application.properties

```
spring.application.name=PagingAndSorting
server.port=6755
```

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.5.6</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.example</groupId>
  <artifactId>PagingAndSortingApplication</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>PagingAndSorting</name>
  <properties>
    <java.version>17</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
</project>
```

Output:

```
Console x Progress
PagingAndSorting - PagingAndSortingApplication [Spring Boot App] C:\Program Files\Java\jdk-17\bin\javaw.exe (02-Nov-2025, 9:39:05 pm elapsed: 0:00:11)

Spring Boot (v3.5.6)
2025-11-02T21:39:08.436+05:30 INFO 19952 --- [PagingAndSorting] main] c.example.PagingAndSortingApplic
2025-11-02T21:39:08.444+05:30 INFO 19952 --- [PagingAndSorting] main] c.example.PagingAndSortingApplic
2025-11-02T21:39:10.075+05:30 INFO 19952 --- [PagingAndSorting] main] o.s.b.w.embedded.tomcat.TomcatW
2025-11-02T21:39:10.112+05:30 INFO 19952 --- [PagingAndSorting] main] o.apache.catalina.core.StandardS
2025-11-02T21:39:10.113+05:30 INFO 19952 --- [PagingAndSorting] main] o.apache.catalina.core.StandardS
2025-11-02T21:39:10.237+05:30 INFO 19952 --- [PagingAndSorting] main] o.a.c.c.C.[Tomcat].[localhost].(
2025-11-02T21:39:10.239+05:30 INFO 19952 --- [PagingAndSorting] main] w.s.c.ServletWebServerApplicatio
2025-11-02T21:39:11.235+05:30 INFO 19952 --- [PagingAndSorting] main] o.s.b.w.embedded.tomcat.TomcatW
2025-11-02T21:39:11.259+05:30 INFO 19952 --- [PagingAndSorting] main] c.example.PagingAndSortingApplic
```

```
localhost:6755/products?page=0&size=4
Pretty-print
[{"id":1,"name":"Laptop","price":53000.0}, {"id":2,"name":"Mobile","price":20000.0}, {"id":3,"name":"Tablet","price":15000.0}, {"id":4,"name":"Headphones","price":3000.0}]
```

```
localhost:6755/products?page=1&size=4
Pretty-print
[{"id":5,"name":"SmartWatch","price":8000.0}, {"id":6,"name":"Camera","price":25000.0}, {"id":7,"name":"Monitor","price":12000.0}]
```

```
localhost:6755/products?page=2&size=4
Pretty-print
[]
```

12) Aim: SOAP-Based Web Service

Description:

In this experiment, we will create a SOAP-based Web Service using Spring Boot and Spring Web Services. The application exposes a simple SOAP endpoint that receives a user's name and returns a personalized greeting message in XML format. **Entity** – Product with id, name, and price.

- **SoapExampleApplication.java** – Main class to run the Spring Boot application.
- **WebServiceConfig.java** – Configures the SOAP servlet, WSDL endpoint, and schema.
- **HelloEndpoint.java** – Handles SOAP requests and generates responses.
- **HelloRequest.java** and **HelloResponse.java** – JAXB-annotated classes for request and response objects.
- **Hello.xsd** – Defines the XML schema for SOAP messages.
- **application.properties** – Contains application name and port configuration.
- **pom.xml** – Maven configuration file for dependencies.

Program:

Hello.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://example.com/soap"
  xmlns:tns="http://example.com/soap"
  elementFormDefault="qualified">
  <xs:element name="helloRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="helloResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="message" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

HelloEndpoint.java

```
package com.example;

import com.example.HelloRequest;
import com.example.HelloResponse;
import org.springframework.ws.server.endpoint.annotation.Endpoint;
import org.springframework.ws.server.endpoint.annotation.PayloadRoot;
import org.springframework.ws.server.endpoint.annotation.RequestPayload;
import org.springframework.ws.server.endpoint.annotation.ResponsePayload;

@Endpoint
public class HelloEndpoint {
    private static final String NAMESPACE = "http://example.com/soap";

    @PayloadRoot(namespace = NAMESPACE, localPart = "helloRequest")
    @ResponsePayload
    public HelloResponse sayHello(@RequestPayload HelloRequest request) {
        HelloResponse response = new HelloResponse();
        response.setMessage("Hello, " + request.getName() + "! Welcome to Spring Boot SOAP.");
        return response;
    }
}
```

HelloRequest.java

```
package com.example;

import jakarta.xml.bind.annotation.XmlElement;
import jakarta.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name = "helloRequest", namespace = "http://example.com/soap")
public class HelloRequest {
    private String name;

    @XmlElement
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}
```

HelloResponse.java

```
package com.example;

import jakarta.xml.bind.annotation.XmlElement;
import jakarta.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name = "helloResponse", namespace = "http://example.com/soap")
public class HelloResponse {
    private String message;

    @XmlElement
    public String getMessage() { return message; }
    public void setMessage(String message) { this.message = message; }
}
```

application.properties

```
spring.application.name=SoapExample
server.port=6756
```

ProductController.java

```
package com.example.demo;

import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
@RequestMapping("/products")
public class ProductController {
    private final ProductService service;
    public ProductController(ProductService service) {
        this.service = service;
    }

    @PostMapping("/add")
    public Product addProduct(@RequestBody Product product) {
        return service.saveProduct(product);
    }

    @GetMapping("/all")
    public List<Product> getAllProducts() {
        return service.getAllProducts();
    }
}
```

WebServiceConfig.java

```
package com.example;
```

```
import org.springframework.boot.web.servlet.ServletRegistrationBean;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.io.ClassPathResource;
import org.springframework.ws.config.annotation.EnableWs;
import org.springframework.ws.transport.http.MessageDispatcherServlet;
import org.springframework.ws.wsdl.wsdl11.DefaultWsdl11Definition;
import org.springframework.xml.xsd.SimpleXsdSchema;
import org.springframework.xml.xsd.XsdSchema;
```

```
@EnableWs
```

```
@Configuration
```

```
public class WebServiceConfig {
```

```
    @Bean
```

```
    public ServletRegistrationBean<MessageDispatcherServlet> messageDispatcherServlet
        (ApplicationContext ctx)
    {
        MessageDispatcherServlet servlet = new MessageDispatcherServlet();
        servlet.setApplicationContext(ctx);
        servlet.setTransformWsdlLocations(true);
        return new ServletRegistrationBean<>(servlet, "/ws/*");
    }
```

```
    @Bean(name = "hello")
```

```
    public DefaultWsdl11Definition defaultWsdl11Definition(XsdSchema helloSchema) {
        DefaultWsdl11Definition definition = new DefaultWsdl11Definition();
        definition.setPortTypeName("HelloPort");
        definition.setLocationUri("/ws");
        definition.setTargetNamespace("http://example.com/soap");
        definition.setSchema(helloSchema);
        return definition;
    }
```

```
    @Bean
```

```
    public XsdSchema helloSchema() {
        return new SimpleXsdSchema(new ClassPathResource("hello.xsd"));
    }
}
```

SoapExampleApplication.java

```
package com.example;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SoapExampleApplication {

    public static void main(String[] args) {
        SpringApplication.run(SoapExampleApplication.class, args);
    }

}
```

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.5.6</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.example</groupId>
  <artifactId>SoapExampleApplication</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>SoapExample</name>
  <properties>
    <java.version>17</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
```

```

        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web-services</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <!-- JAXB for XML binding -->
    <dependency>
        <groupId>jakarta.xml.bind</groupId>
        <artifactId>jakarta.xml.bind-api</artifactId>
    </dependency>
    <dependency>
        <groupId>org.glassfish.jaxb</groupId>
        <artifactId>jaxb-runtime</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.ws</groupId>
        <artifactId>spring-ws-core</artifactId>
    </dependency>
    <dependency>
        <groupId>wsdl4j</groupId>
        <artifactId>wsdl4j</artifactId>
    </dependency>
</dependencies>
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
</project>

```

Output:

[illegible]