in28minutes

Master Java Web Services and REST API with Spring Boot

Learn to develop RESTful and SOAP Java Web Services with Spring and Spring Boot in 90 easy steps

Table of Contents

Congratulations

You have made a great choice in learning with in28Minutes. You are joining 150,000+ Learners learning everyday with us.

150,000+ Java beginners are learning from in28Minutes to become experts on APIs, Web Services and Microservices with Spring, Spring Boot and Spring Cloud.

About in28Minutes

How did in28Minutes get to 100,000 learners across the world?

Total Students

115,263

Top Student Locations	
United States	27%
India	22%
Poland	3%
United Kingdom	3%
Canada	2%

Countries With Students

181

We are focused on creating the awesome course (learning) experiences. Period.

An awesome learning experience?

What's that?

You need to get insight into the in28Minutes world to answer that.

You need to understand "***The in28Minutes Way***"

- What are our beliefs?
- What do we love?
- Why do we do what we do?
- How do we design our courses?

Let's get started on "***The in28Minutes Way***"!

Important Components of "The in28Minutes Way"

- Continuous Learning
- Hands-on
- We don't teach frameworks. We teach building applications!
- We want you to be strong on the fundamentals

- Step By Step

- Efficient and Effective
- Real Project Experiences
- Debugging and Troubleshooting skills
- Modules - Beginners and Experts!
- Focus on Unit Testing
- Code on Github

- Design and Architecture
- Modern Development Practices
- Interview Guides
- Bring the technology trends to you
- Building a connect
- Socially Conscious
- We care for our learners
- We love what we do

Troubleshooting Guide

We love all our 100,000 learners. We want to help you in every way possible.

We do not want you to get stuck because of a simple error.

This 50 page troubleshooting guide and faq is our way of thanking you for choosing to learn from in28Minutes.

.in28Minutes Trouble Shooting Guide

Getting Started

Recommended Versions

Tool/Framework/Language	Recommended Version	More Details
Java	Java 8	http://www.in28minutes.com/spr...
Eclipse	Eclipse Java EE Oxygen	Basics
Spring Boot	Spring Boot 2.0.0.RELEASE	
Spring	Version 5 or greater	

Installation

- Video : https://www.youtube.com/playlist?list=PLBBog2r6uMCSmMVTW_QmDLyASBvovyAO3
- PDF : https://github.com/in28minutes/SpringIn28Minutes/blob/master/InstallationGuide-JavaEclipseAndMaven_v2.pdf
- More Details : https://github.com/in28minutes/getting-started-in-5-steps

Troubleshooting

- A 50 page troubleshooting guide with more than 200 Errors and Questions answered

Spring Web Services - Course Overview

Github Repo

https://github.com/in28minutes/spring-web-services

Course Overview

Title	Github Folder
Introduction To Web Services	None
SOAP Web Services with Spring and Spring Boot	Project Folder on Github
RESTful Web Services with Spring and Spring Boot	Project Folder on Github
Connecting RESTful Web Service to JPA	Project Folder on Github
RESTful Web Services - Best Practices	None

3 Bonus Sections - Introduction to Spring, Spring Boot and JPA

Title	Category	Github
Spring Framework in 10 Steps	Introduction	Project Folder on Github
Spring Boot in 10 Steps	Introduction	Project Folder on Github
JPA in 10 Steps	Introduction	Project Folder on Github

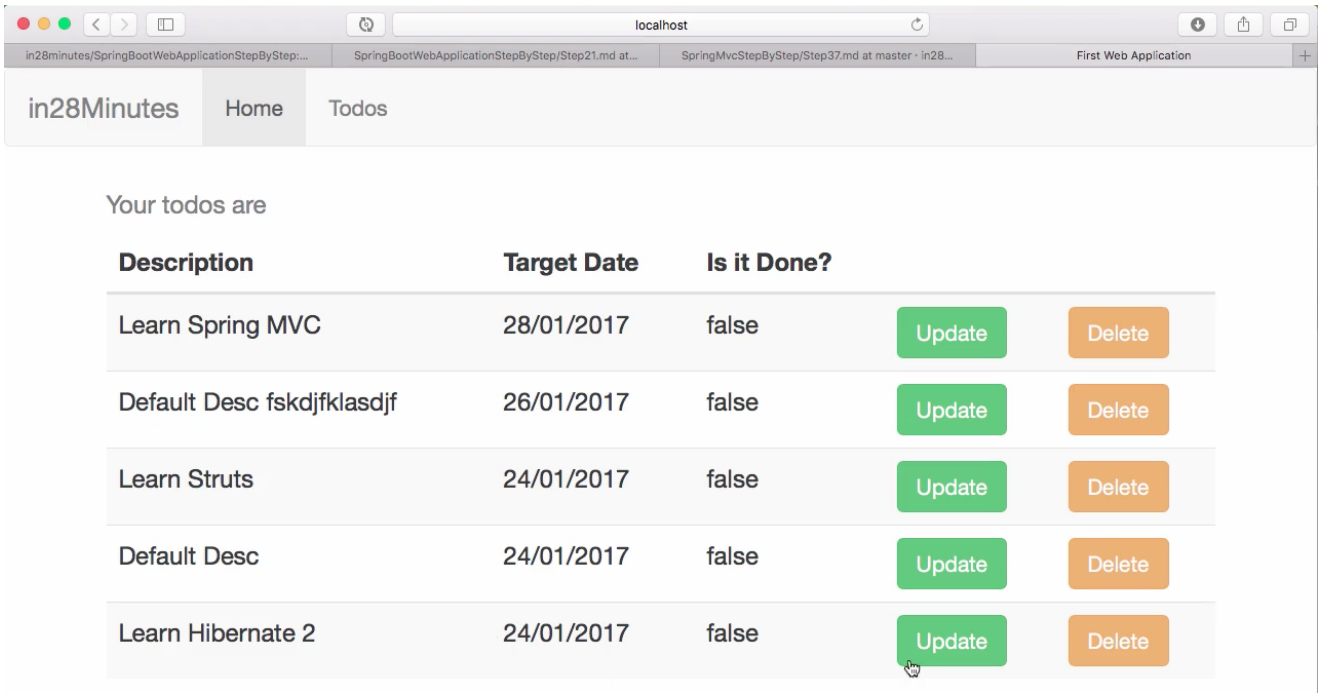
Introduction to Web Services

Introduction to Web Services

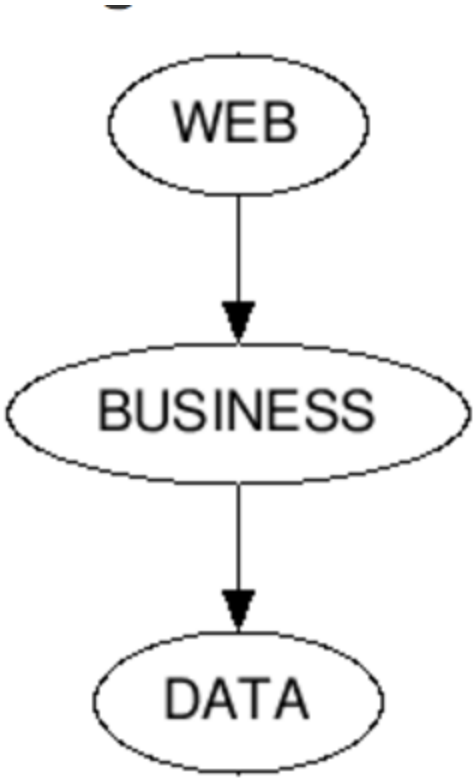
- What is a Web Service?
- Important How Questions related to Web Services
- Web Services - Key Terminology
- Introduction to SOAP Web Services
- Introduction to RESTful Web Services
- SOAP vs RESTful Web Services

Web Service

Service delivered over the web?



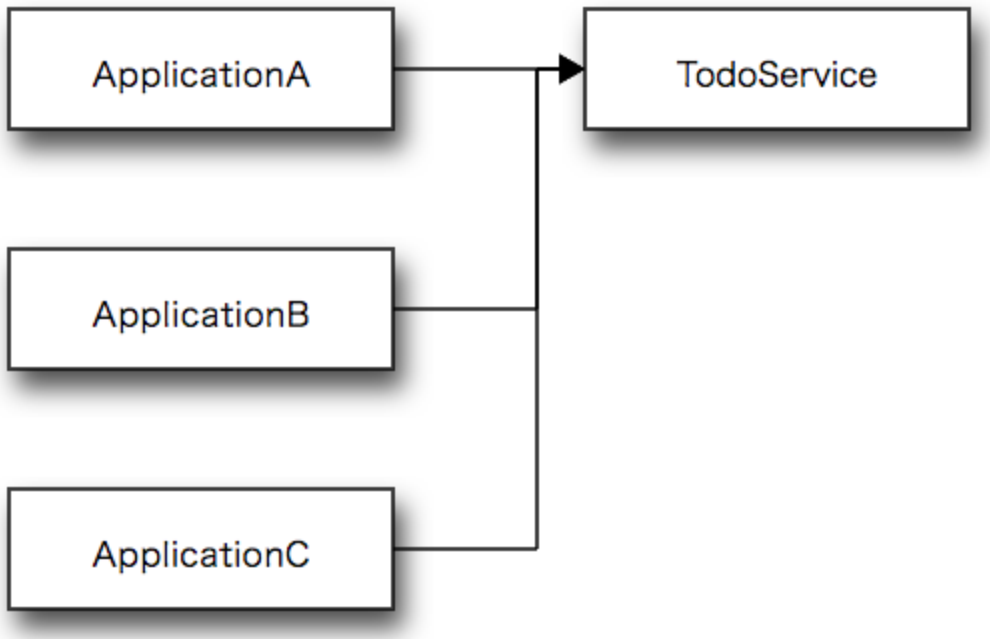
Is the Todo Management Application a Web Service?

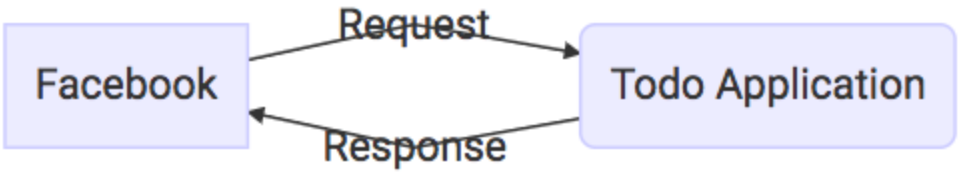
- It delivers HTML output - Not consumable by other applications.
- 

- Can I reuse the Business Layer by creating a JAR?
    - Not Platform independent
    - Communication of Changes
    - Managing Dependencies - like Database

How can I make my Todo application consumable by other applications?

That where we get into the concept of a web service!





Web Service - W3C definition

> *Software system designed to support interoperable machine-to-machine interaction over a network.*

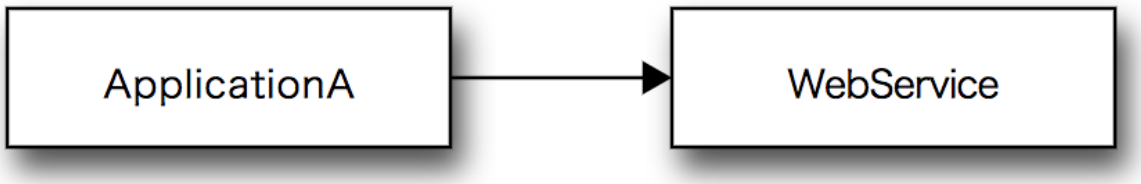
3 Keys

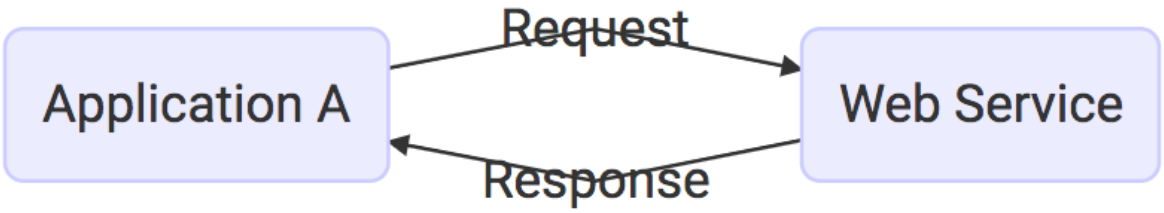
- Designed for machine-to-machine (or application-to-application) interaction

- Should be interoperable - Not platform dependent
- Should allow communication over a network

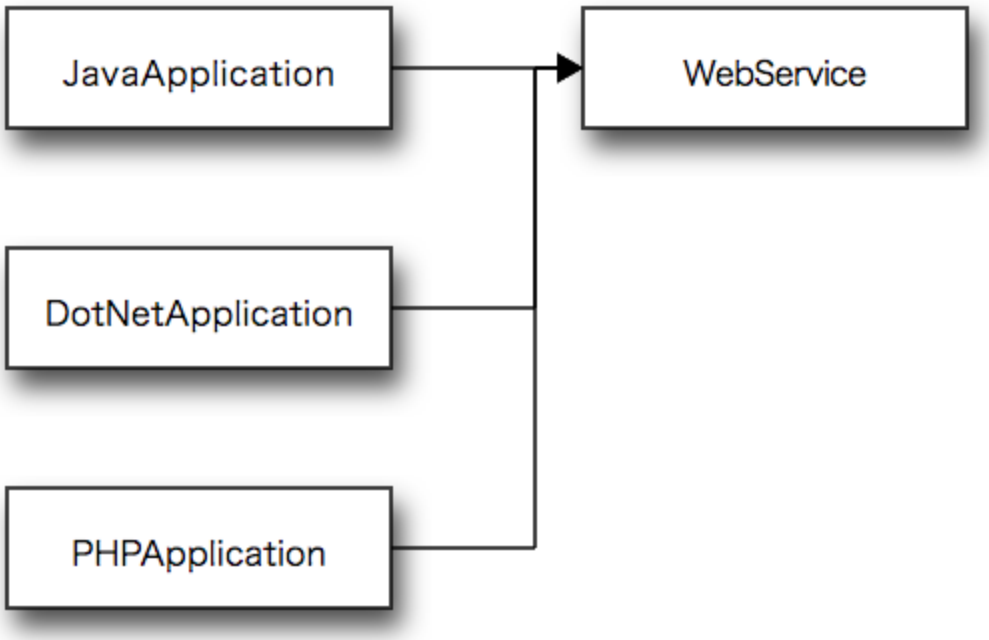
How?

How does data exchange between applications take place?





How can we make web services platform independent?



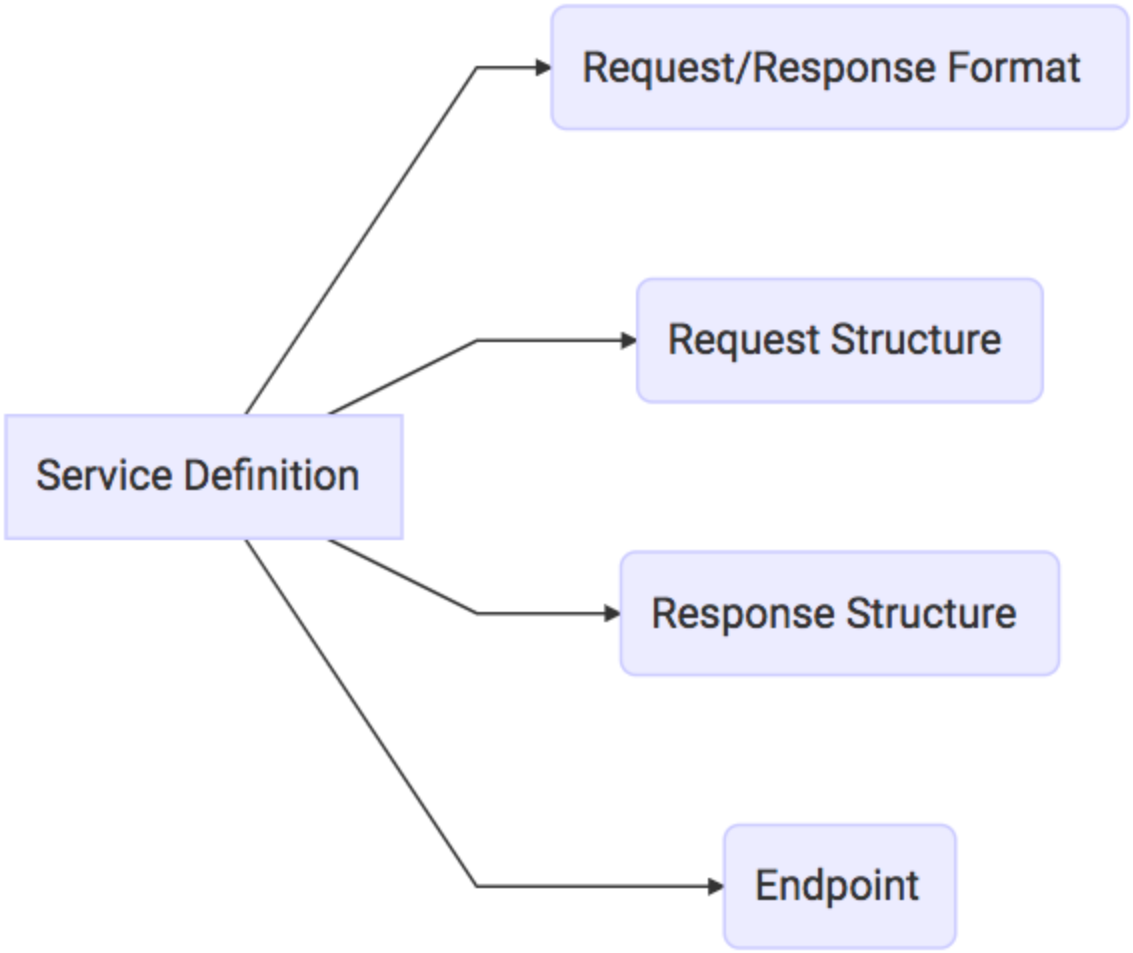
XML

```
        <getCourseDetailsRequest>
            <id>Course1</id>
        </getCourseDetailsRequest>
```

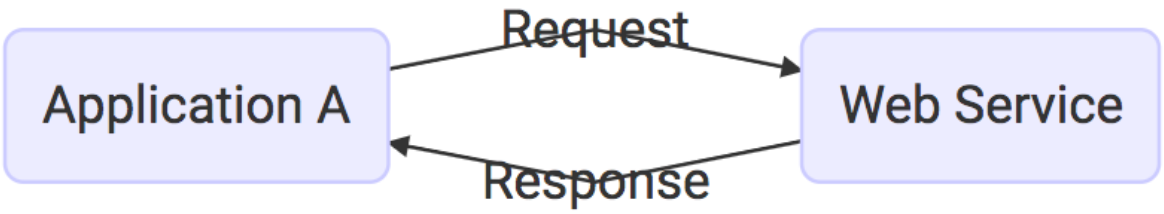
JSON

```
[
  {
    "id": 1,
    "name": "Even",
    "birthDate": "2017-07-10T07:52:48.270+0000"
  },
  {
    "id": 2,
    "name": "Abe",
    "birthDate": "2017-07-10T07:52:48.270+0000"
  }
]
```

How does the Application A know the format of Request and Response?



How does Application A and Web Service convert its internal data to (XML or JSON)?



Key Terminology

- Request and Response

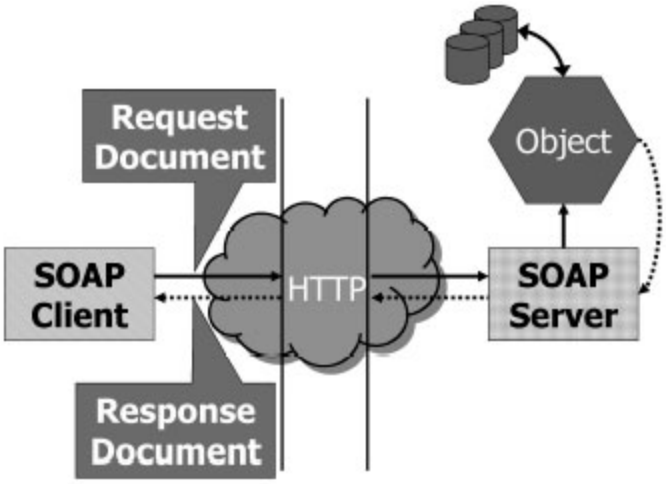
- Message Exchange Format
  - XML and JSON

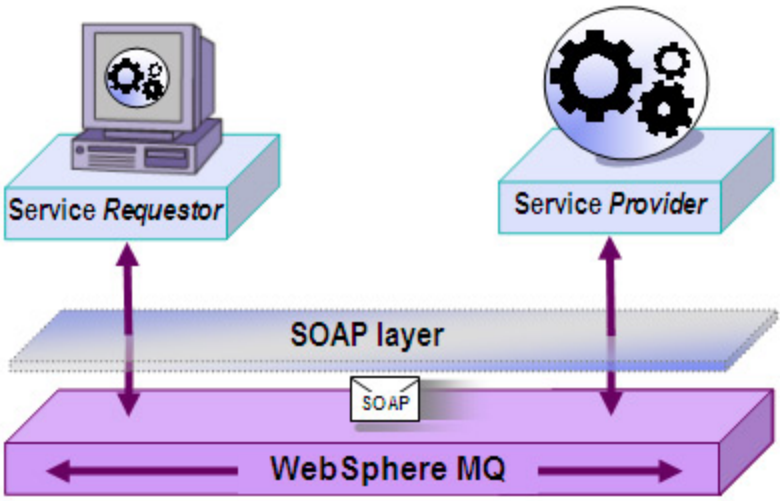
Key Terminology

- Service Provider or Server
- Service Consumer or Client
- Service Definition

Key Terminology

- Transport
  - HTTP and MQ





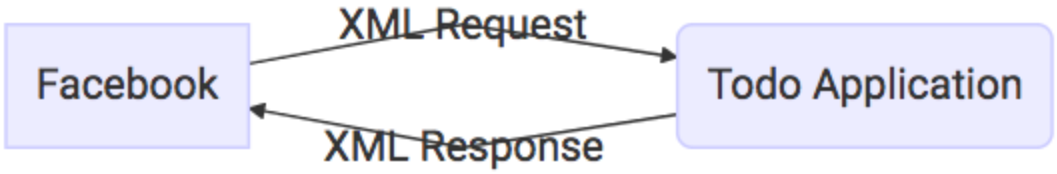
Web Service Groups

- SOAP-based
- REST-styled

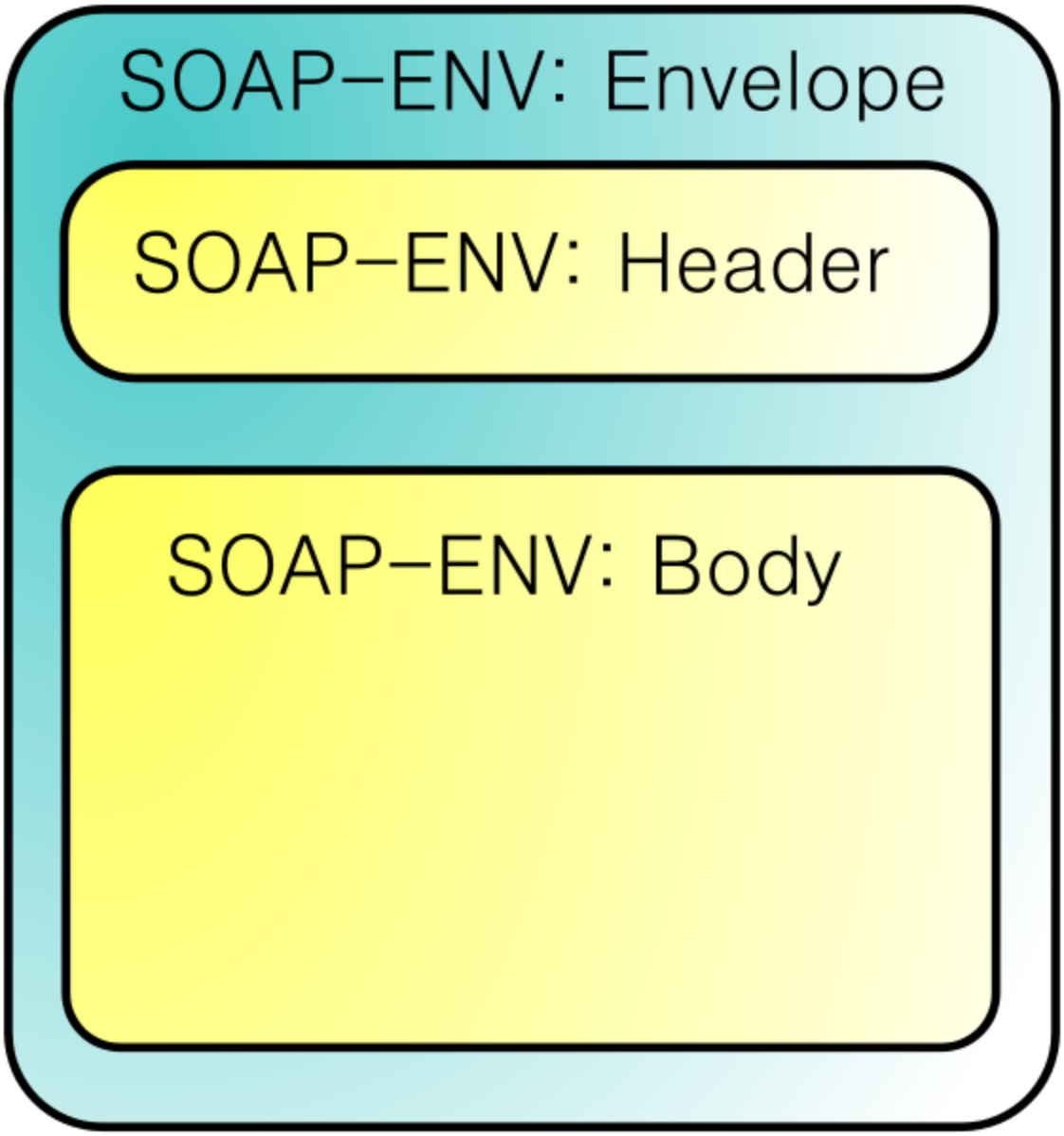
SOAP and REST are not really comparable.

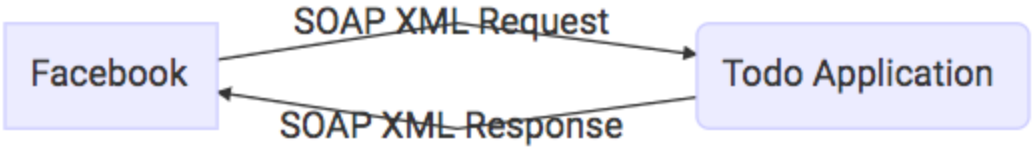
SOAP

SOAP?




```
<getCourseDetailsRequest>
     <id>Course1</id>
</getCourseDetailsRequest>
```






```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <SOAP-ENV:Body>
        <ns2:getCourseDetailsResponse
xmlns:ns2="http://in28minutes.com/courses">
            <ns2:course>
```

```
 <ns2:id>Course1</ns2:id>
                <ns2:name>Spring</ns2:name>
                <ns2:description>10 Steps</ns2:description>
            </ns2:course>
        </ns2:getCourseDetailsResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP

- Format - SOAP XML Request - SOAP XML Response
- Transport
  - SOAP over MQ
  - SOAP over HTTP

- Service Definition
  - WSDL

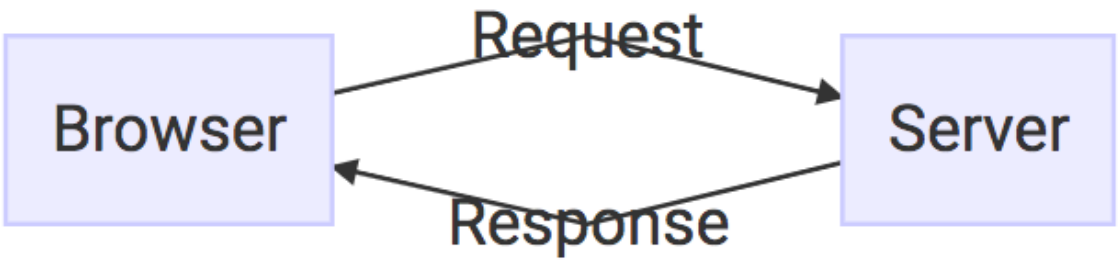
REST

> *REpresentational State Transfer*

> *REST is a style of software architecture for distributed hypermedia systems*

Make best use of HTTP

REST(REpresentational State Transfer)	
HTTP	
HTTP Methods (GET, PUT, POST..)	HTTP Status Codes (200, 404..)



Key abstraction - Resource

- A resource has an URI (Uniform Resource Identifier)
- /users/Ranga/todos/1
- /users/Ranga/todos
- /users/Ranga
- A resource can have different representations
- XML
- HTML
- JSON

Example

- Create a User - POST /users
- Delete a User - DELETE /users/1
- Get all Users - GET /users
- Get one Users - GET /users/1

REST

- Data Exchange Format - No Restriction. JSON is popular
- Transport
  - Only HTTP

- Service Definition
  - No Standard. WADL/Swagger/...

REST vs SOAP

- Restrictions vs Architectural Approach
- Data Exchange Format
- Service Definition
- Transport

-
- Ease of implementation

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <SOAP-ENV:Body>
        <ns2:GetCourseDetailsRequest
xmlns:ns2="http://in28minutes.com/courses">
                <ns2:id>Course1</ns2:id>
         </ns2:GetCourseDetailsRequest>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <SOAP-ENV:Body>
        <ns2:GetCourseDetailsResponse
xmlns:ns2="http://in28minutes.com/courses">
            <ns2:CourseDetails>
                <ns2:id>Course1</ns2:id>
                <ns2:name>Spring</ns2:name>
                <ns2:description>10 Steps</ns2:description>
            </ns2:CourseDetails>
        </ns2:GetCourseDetailsResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Web Services

Github Folder

https://github.com/in28minutes/spring-web-services/tree/master/soap-web-services

SOAP Web Services Step By Step Details

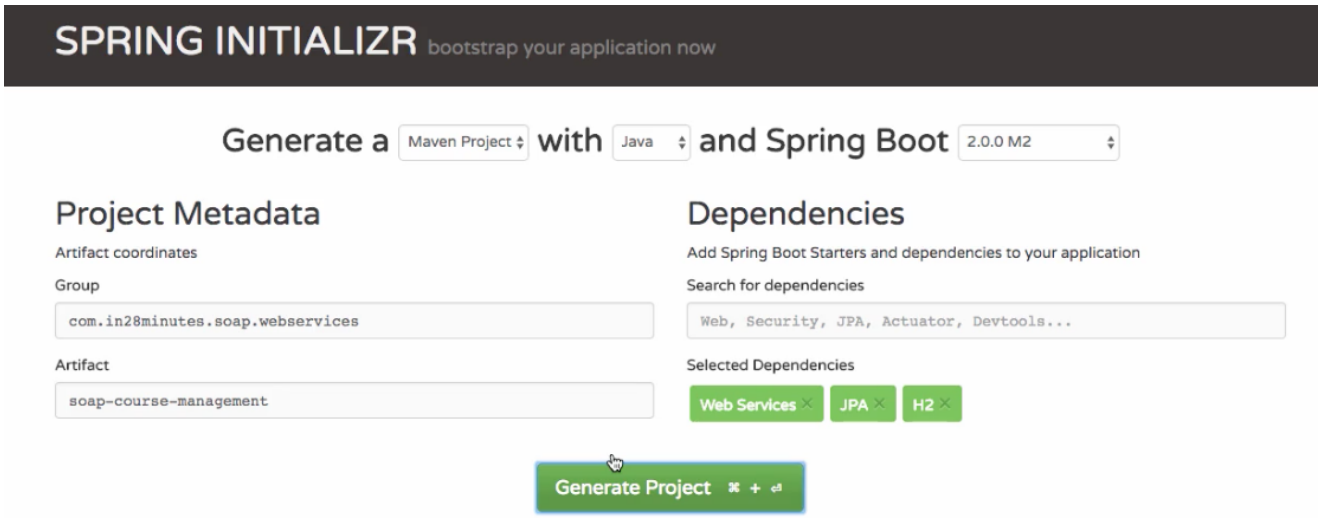
- Step 01 - Initialize a Spring Web Services application with Spring Boot
- Step 02 - Overview of creating SOAP Web Service using Contract First Approach
- Step 03 - Define Request and Response XML Structure
- Step 04 - Define XML Schema Definition (XSD) for Request - GetCourseDetailsRequest
- Step 05 - Define XML Schema Definition (XSD) for Respone - GetCourseDetailsResponse
- Step 06 - More about XML Schema Definition and Implementing XSD Best Practices
- Step 07 - Introduction to Java API for XML Binding (JAXB) and Configuring JAXB 2 Maven Plugin
- Step 08 - Configuring an Endpoint for GetCourseDetailsRequest
- Step 09 - Spring Web Services Configuration - Message Dispatcher Servlet
- Step 10 - Spring Web Services Configuration - Generating WSDL
- Step 11 - Using Wizdler to execute SOAP Requests
- Step 12 - Implementing a service - Course Details Service - backend with in memory array list
- Step 13 - Implementing SOAP Web Service for GetAllCourseDetailsRequest
- Step 14 - Quick introduction to different parts of a WSDL
- Step 15 - Implementing SOAP Web Service for DeleteCourseDetailsRequest
- Step 16 - Improving the DeleteCourseDetailsRequest - Using an Enum for Status
- Step 17 - Exception Handling and SOAP Fault Responses

- Step 18 - Implementing Security for SOAP Web Services with WS Security

Step 01 - Initialize a Spring Web Services application with Spring Boot

Creating a Spring Project with Spring Initializr is a cake walk.

> *Spring Initializr http://start.spring.io/ is great tool to bootstrap your Spring Boot projects.*



- Launch Spring Initializr and choose the following
  - Choose Version 2.0.0.RELEASE or greater
  - Choose Group as shown in the figure
  - Choose Artifact as shown in the figure
  - Choose Dependencies as shown in the figure

- Click Generate Project.
- Import the project into Eclipse.

/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>

        <groupId>com.in28minutes.soap.webservices</groupId>
        <artifactId>soap-course-management</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>soap-course-management</name>
        <description>Demo project for Spring
Boot</description>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>2.0.0.RELEASE</version>
                <relativePath/> <!-- lookup parent from
repository -->
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
                <java.version>1.8</java.version>
        </properties>
```

```
        <dependencies>


        <dependency>

<groupId>org.springframework.boot</groupId>


        <artifactId>spring-boot-starter-data-
jpa</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web-services</artifactId>
                </dependency>

                <dependency>
                        <groupId>com.h2database</groupId>
                        <artifactId>h2</artifactId>
                        <scope>runtime</scope>
                </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>
```

/src/main/java/com/in28minutes/soap/webservices/soapcoursemanagement/SoapCourseManagementApplication.java

```
package
com.in28minutes.soap.webservices.soapcoursemanagement;
 import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplicatio
n;
 @SpringBootApplication
public class SoapCourseManagementApplication {

        public static void main(String[] args) {

SpringApplication.run(SoapCourseManagementApplication.class
, args);
        } }
```

/src/main/resources/application.properties

/src/test/java/com/in28minutes/soap/webservices/soapcoursemanagement/SoapCourseManagementApplicationTests.java

```
package
com.in28minutes.soap.webservices.soapcoursemanagement;
 import org.junit.Test; import org.junit.runner.RunWith;
import
org.springframework.boot.test.context.SpringBootTest;
import
```

```
org.springframework.test.context.junit4.SpringRunner;
  @RunWith(SpringRunner.class) @SpringBootTest public class
```

```
 SoapCourseManagementApplicationTests {

        @Test
        public void contextLoads()  {
        }

}
```

Step 02 - Overview of creating SOAP Web Service using Contract First Approach

Lets first define an XSD.

Step 03 - Define Request and Response XML Structure

/example-files/Request.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<GetCourseDetailsRequest
xmlns="http://in28minutes.com/courses"
xsi:schemaLocation="http://in28minutes.com/courses course-
details.xsd"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
        <id>123</id> <!-- numbers  -->
</GetCourseDetailsRequest>
```

/example-files/Response.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<GetCourseDetailsResponse
xmlns="http://in28minutes.com/courses">
        <CourseDetails>
                <id>123</id>
                <name>Spring in28minutes</name>
                <description>You would learn the basics of
Spring  Framework</description>
        </CourseDetails> </GetCourseDetailsResponse>
```

Step 04 - Define XML Schema Definition (XSD) for Request - GetCourseDetailsRequest

/example-files/course-details.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<schema xmlns="http://www.w3.org/2001/XMLSchema"

 targetNamespace="http://in28minutes.com/courses"
xmlns:tns="http://in28minutes.com/courses"
elementFormDefault="qualified">
        <element name="GetCourseDetailsRequest">
                <complexType>
                        <sequence>
                                <element name= "id"
type="integer"></element>
                        </sequence>
                </complexType>
        </element>
</schema>

<!--
<GetCourseDetailsRequest
xmlns="http://in28minutes.com/courses">
        <id>123</id>
</GetCourseDetailsRequest>
 -->
```

Step 05 - Define XML Schema Definition (XSD) for Respone - GetCourseDetailsResponse

/example-files/Response.xml Modified

```
<?xml version="1.0" encoding="UTF-8"?>
<GetCourseDetailsResponse
xmlns="http://in28minutes.com/courses"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:schemaLocation="http://in28minutes.com/courses  course-
details.xsd">
        <CourseDetails>
                <id>123</id>
                <name>Spring in28minutes</name>
                <description>You would learn the basics of
Spring Framework</description>

        </CourseDetails> </GetCourseDetailsResponse>
```

/example-files/course-details.xsd Modified

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://in28minutes.com/courses"
xmlns:tns="http://in28minutes.com/courses"
elementFormDefault="qualified">

        <element name="GetCourseDetailsRequest">
                <complexType>
                        <sequence>
                                <element name= "id"
type="integer"></element>
                        </sequence>
                </complexType>
        </element>

        <element name="GetCourseDetailsResponse">
                <complexType>
                        <sequence>
                                <element name=
"CourseDetails" type="tns:CourseDetails"></element>
                        </sequence>
                </complexType>
        </element>

        <complexType
```

```
 name="CourseDetails">
                <sequence>
                        <element name="id"
type="integer"/>
                        <element name="name"
type="string"/>
                        <element name="description"
type="string"/>


        </sequence>
        </complexType>
         </schema>
```

Step 06 - More about XML Schema Definition and Implementing XSD Best Practices

/example-files/course-details.xsd Modified

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://in28minutes.com/courses"
xmlns:tns="http://in28minutes.com/courses"
elementFormDefault="qualified">

        <xs:element name="GetCourseDetailsRequest">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name= "id"
type="xs:integer"/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>

        <xs:element  name="GetCourseDetailsResponse">
```

```
        <xs:complexType>
                        <xs:sequence>
                                <xs:element name=
"CourseDetails" type="tns:CourseDetails"/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>

        <xs:complexType name="CourseDetails">
                <xs:sequence>
                        <xs:element name="id"



 type="xs:integer"/>
                        <xs:element name="name"
type="xs:string"/>
                        <xs:element name="description"
type="xs:string"/>
                </xs:sequence>
        </xs:complexType>

</xs:schema>
```

Step 07 - Introduction to Java API for XML Binding (JAXB) and Configuring JAXB 2 Maven Plugin

Step 08 - Configuring an Endpoint for GetCourseDetailsRequest

/pom.xml Modified

New Lines

```
<plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>jaxb2-maven-plugin</artifactId>
```

```
        <version>1.6</version>
        <executions>
                <execution>
                        <id>xjc</id>
                        <goals>
                                <goal>xjc</goal>
                        </goals>
                </execution>
        </executions>
        <configuration>

<schemaDirectory>${project.basedir}/src/main/resources</sch
emaDirectory>



        <outputDirectory>${project.basedir}/src/main/java</
outputDirectory>
                <clearOutputDir>false</clearOutputDir>
        </configuration> </plugin>
```

/src/main/java/com/in28minutes/soap/webservices/soapcoursemanagement/soap/CourseDetailsEndpoint.java New

```
package
com.in28minutes.soap.webservices.soapcoursemanagement.soap;

import
org.springframework.ws.server.endpoint.annotation.Endpoint;
import
org.springframework.ws.server.endpoint.annotation.PayloadRo
ot;
import
org.springframework.ws.server.endpoint.annotation.RequestPa
yload;
import
org.springframework.ws.server.endpoint.annotation.ResponseP
ayload;

import com.in28minutes.courses.CourseDetails; import
```

```
com.in28minutes.courses.GetCourseDetailsRequest;
import com.in28minutes.courses.GetCourseDetailsResponse;
```

```
@Endpoint public class CourseDetailsEndpoint {

        // method
        // input - GetCourseDetailsRequest
        // output - GetCourseDetailsResponse

        // http://in28minutes.com/courses
        // GetCourseDetailsRequest
        @PayloadRoot(namespace =
"http://in28minutes.com/courses", localPart =
"GetCourseDetailsRequest")
        @ResponsePayload
        public GetCourseDetailsResponse

 processCourseDetailsRequest(@RequestPayload
GetCourseDetailsRequest request)

 {
                GetCourseDetailsResponse response = new
GetCourseDetailsResponse();

                CourseDetails courseDetails = new
CourseDetails();
                courseDetails.setId(request.getId());
                courseDetails.setName("Microservices
Course");
                courseDetails.setDescription("That would be
a wonderful  course!");

                response.setCourseDetails(courseDetails);

                return response;
        }

}
```

/src/main/resources/course-details.xsd New

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://in28minutes.com/courses"
xmlns:tns="http://in28minutes.com/courses"
elementFormDefault="qualified">

        <xs:element  name="GetCourseDetailsRequest">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name= "id"
type="xs:int"/>


        </xs:sequence>
                </xs:complexType>
        </xs:element>


        <xs:element name="GetCourseDetailsResponse">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name=
"CourseDetails"  type="tns:CourseDetails"/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>

        <xs:complexType  name="CourseDetails">
                <xs:sequence>
                        <xs:element name="id"
type="xs:int"/>
                        <xs:element name="name"
type="xs:string"/>
                        <xs:element name="description"
type="xs:string"/>
                </xs:sequence>
        </xs:complexType>
```

```
</xs:schema>
```

Step 09 - Spring Web Services Configuration - Message Dispatcher Servlet

Step 10 - Spring Web Services Configuration - Generating WSDL

/pom.xml Modified

New Lines

```
        <dependency>
                        <groupId>wsdl4j</groupId>
                        <artifactId>wsdl4j</artifactId>
                </dependency>
```

/src/main/java/com/in28minutes/soap/webservices/soapcoursemanagement/soap/WebServiceConfig.java New

```
package
com.in28minutes.soap.webservices.soapcoursemanagement.soap;
  import
org.springframework.boot.web.servlet.ServletRegistrationBea
n;
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.MessageDispatcherServ
let; import
org.springframework.ws.wsdl.wsdl11.DefaultWsdl11Definition;
import org.springframework.xml.xsd.SimpleXsdSchema;
import org.springframework.xml.xsd.XsdSchema;
```

```
//Enable Spring Web Services
@EnableWs
// Spring Configuration
@Configuration
public class WebServiceConfig  {
        // MessageDispatcherServlet
        // ApplicationContext
        // url -> /ws/*

        @Bean
        public ServletRegistrationBean
messageDispatcherServlet(ApplicationContext context)

 {
                MessageDispatcherServlet
messageDispatcherServlet = new MessageDispatcherServlet();

messageDispatcherServlet.setApplicationContext(context);



        messageDispatcherServlet.setTransformWsdlLocations(
true);
                return new
ServletRegistrationBean(messageDispatcherServlet, "/ws/*");
        }

        // /ws/courses.wsdl
        // course-details.xsd
        @Bean(name = "courses")
        public DefaultWsdl11Definition
defaultWsdl11Definition(XsdSchema coursesSchema)  {
                DefaultWsdl11Definition definition = new
DefaultWsdl11Definition();
                definition.setPortTypeName("CoursePort");
```

```
        definition.setTargetNamespace("http://in28minutes.c
om/courses");
                definition.setLocationUri("/ws");
                definition.setSchema(coursesSchema);
                return definition;
        }

        @Bean
        public XsdSchema coursesSchema() {
                return new SimpleXsdSchema(new
ClassPathResource("course-details.xsd"));
        }
}
```

Step 11 - Using Wizdler to execute SOAP Requests

Step 12 - Implementing a service - Course Details Service - backend with in memory array list

/pom.xml Modified

New Lines

```
        <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-
devtools</artifactId>
                </dependency>
```

/src/main/java/com/in28minutes/soap/webservices/soapcoursemanagement/soap/CourseDetailsEndpoint.java Modified

```
@Endpoint public class CourseDetailsEndpoint {

        @Autowired
```

```
        CourseDetailsService service;

        // method
        // input - GetCourseDetailsRequest
        // output - GetCourseDetailsResponse

        // http://in28minutes.com/courses
        // GetCourseDetailsRequest
        @PayloadRoot(namespace =
"http://in28minutes.com/courses", localPart =
"GetCourseDetailsRequest")
        @ResponsePayload
        public GetCourseDetailsResponse
processCourseDetailsRequest(@RequestPayload
GetCourseDetailsRequest request)

 {

                Course course =
service.findById(request.getId());

                return mapCourse(course);
        }

        private GetCourseDetailsResponse mapCourse(Course
course)

 {
                GetCourseDetailsResponse response = new

 GetCourseDetailsResponse();

                CourseDetails courseDetails = new
CourseDetails();

                courseDetails.setId(course.getId());
```

```
        courseDetails.setName(course.getName());


courseDetails.setDescription(course.getDescription());

                response.setCourseDetails(courseDetails);

                return response;
        }
 }
```

/src/main/java/com/in28minutes/soap/webservices/soapcoursemanagement/soap/bean/Course.java New

```
package
com.in28minutes.soap.webservices.soapcoursemanagement.soap.
bean;

public class Course {
        private int id;
        private String name;
        private String  description;


        public Course(int id, String name, String
description) {


        super();
                this.id = id;
                this.name = name;
                this.description =

 description;
        }

        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 String getDescription() {
                return description;
        }


        public void setDescription(String description) {
                this.description = description;
        }

        @Override
        public String toString() {
                return String.format("Course [id=%s,
name=%s, description=%s]", id, name, description);

        }
  }
```

/src/main/java/com/in28minutes/soap/webservices/soapcoursemanagement/soap/service/CourseDetailsService.java New

```
package
com.in28minutes.soap.webservices.soapcoursemanagement.soap.
service;
  import java.util.ArrayList; import java.util.Iterator;
import java.util.List;
```

```
 import org.springframework.stereotype.Component;
 import
com.in28minutes.soap.webservices.soapcoursemanagement.soap.
bean.Course;
 @Component
public class CourseDetailsService {

        private static List<Course> courses = new
ArrayList<>();

        static  {
                Course course1 = new Course(1, "Spring",
"10 Steps");
                courses.add(course1);

                Course course2 = new Course(2, "Spring
MVC", "10 Examples");


        courses.add(course2);

                Course course3 = new Course(3, "Spring
Boot", "6K  Students");
                courses.add(course3);

                Course course4 = new Course(4, "Maven",
"Most popular maven course on internet!");
                courses.add(course4);
        }


        // course - 1
        public Course findById(int id)  {
                for (Course course : courses) {
                        if (course.getId() == id)
                                return
```

```
 course;
                }
                return  null;
        }

        // courses
        public List<Course> findAll() {
                return courses;
        }

        public int deleteById(int id) {
                Iterator<Course> iterator =
courses.iterator();
                while (iterator.hasNext()) {
                        Course course = iterator.next();
                        if (course.getId() == id) {
                                iterator.remove();
                                return 1;


        }
                }
                return 0;
        }

        // updating course & new course }
```

Step 13 - Implementing SOAP Web Service for GetAllCourseDetailsRequest

/src/main/java/com/in28minutes/soap/webservices/soapcoursemanagement/soap/CourseDetailsEndpoint.java Modified

New Lines

```
@Endpoint public class CourseDetailsEndpoint
```

```
 {

        @Autowired
        CourseDetailsService

 service;

        //  method
        // input - GetCourseDetailsRequest
        // output - GetCourseDetailsResponse

        //  http://in28minutes.com/courses
        // GetCourseDetailsRequest
        @PayloadRoot(namespace =
"http://in28minutes.com/courses", localPart =
"GetCourseDetailsRequest")
        @ResponsePayload
        public GetCourseDetailsResponse
processCourseDetailsRequest(@RequestPayload
GetCourseDetailsRequest request) {

                Course course =

 service.findById(request.getId());

                return mapCourseDetails(course);
        }

        private GetCourseDetailsResponse
mapCourseDetails(Course course) {
                GetCourseDetailsResponse response = new
GetCourseDetailsResponse();

response.setCourseDetails(mapCourse(course));
                return response;
        }

        private GetAllCourseDetailsResponse
```

```
 mapAllCourseDetails(List<Course> courses) {
                GetAllCourseDetailsResponse response = new
GetAllCourseDetailsResponse();
                for (Course course : courses) {
                        CourseDetails mapCourse =

 mapCourse(course);

response.getCourseDetails().add(mapCourse);
                }
                return response;
        }

        private CourseDetails mapCourse(Course course)  {
                CourseDetails courseDetails = new
CourseDetails();

                courseDetails.setId(course.getId());



        courseDetails.setName(course.getName());


courseDetails.setDescription(course.getDescription());
                return courseDetails;
        }

        @PayloadRoot(namespace =
"http://in28minutes.com/courses", localPart =
"GetAllCourseDetailsRequest")
        @ResponsePayload
        public GetAllCourseDetailsResponse
processAllCourseDetailsRequest(
                        @RequestPayload
GetAllCourseDetailsRequest request) {

                List<Course> courses =
```

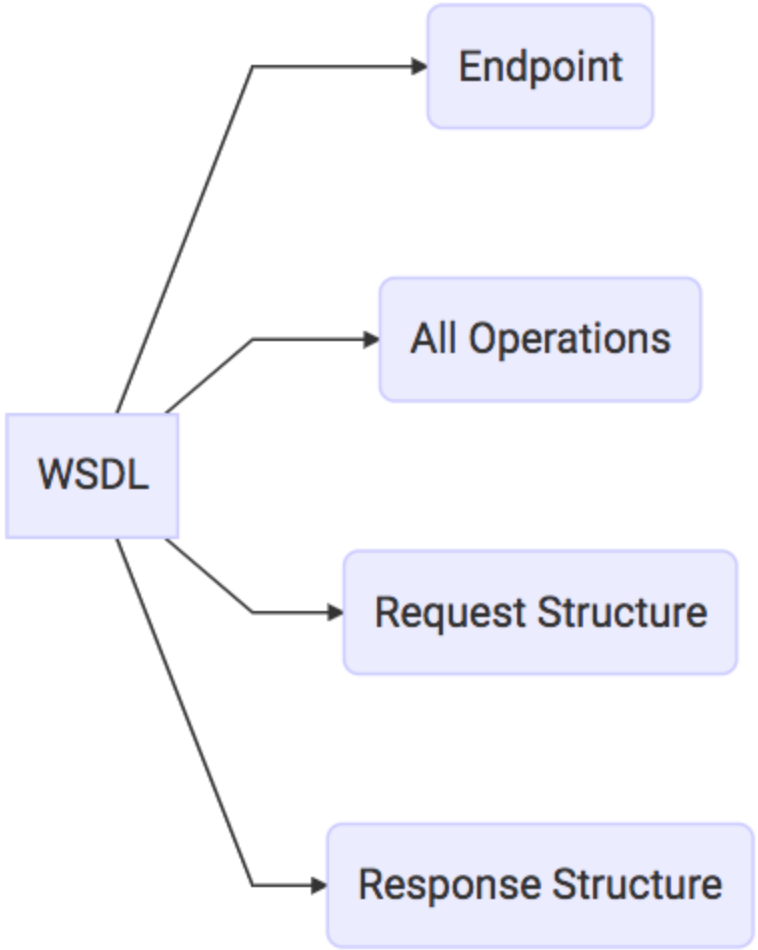
```
 service.findAll();

                return mapAllCourseDetails(courses);
        }
 }
```

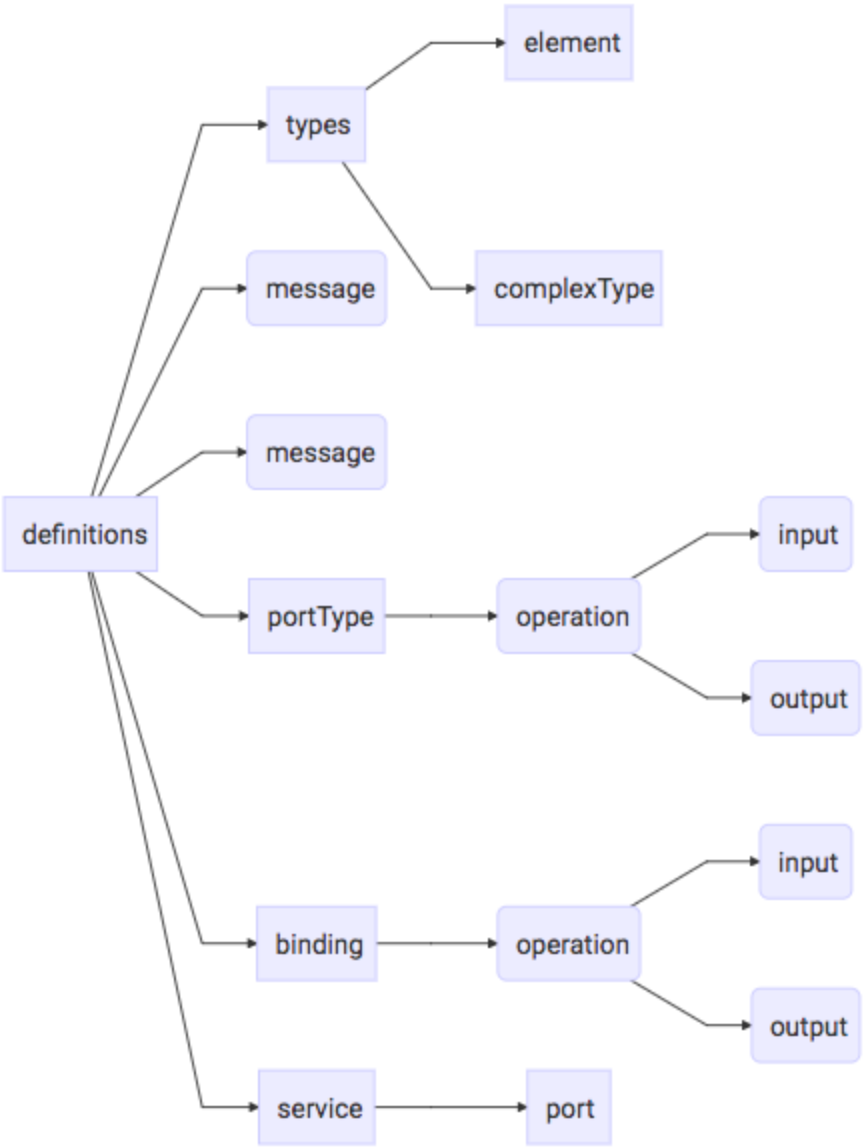
/src/main/resources/course-details.xsd Modified

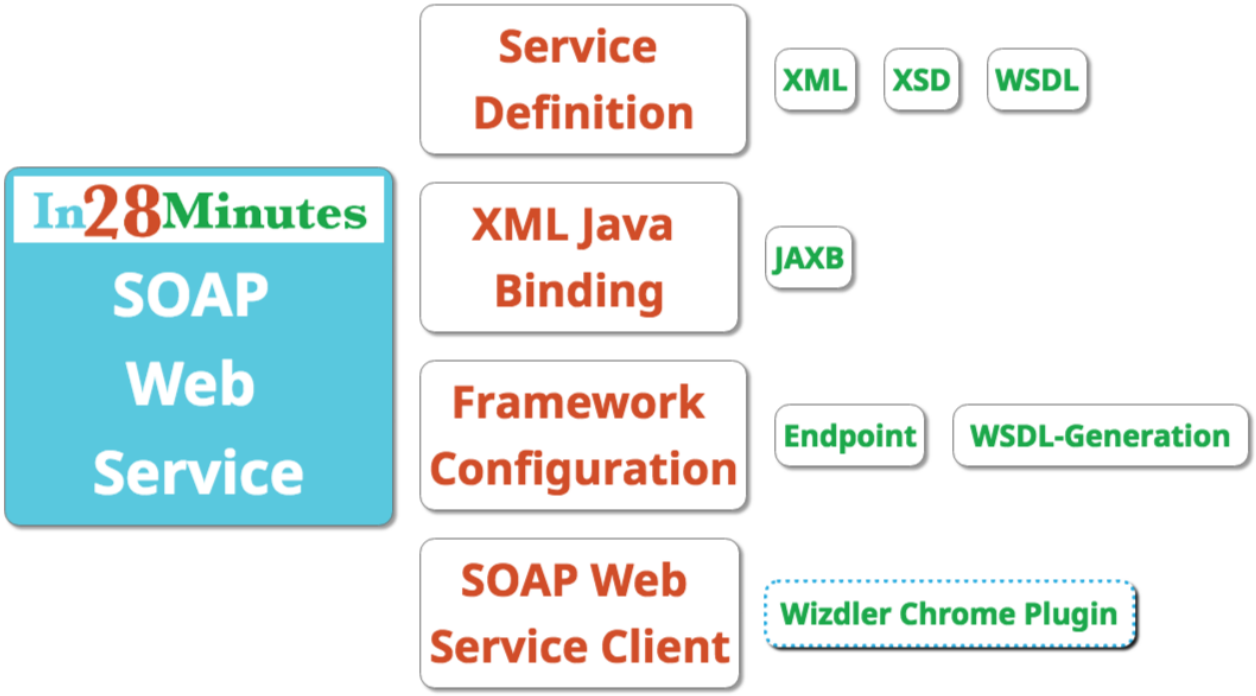
New Lines

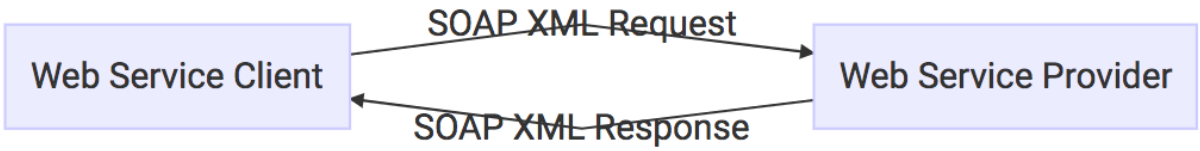
```
<xs:element   name="GetAllCourseDetailsResponse">
        <xs:complexType>
                <xs:sequence>
                        <xs:element name="CourseDetails"
type="tns:CourseDetails"

maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>


<xs:complexType name="CourseDetails">
        <xs:sequence>
                <xs:element name="id" type="xs:int"/>
                <xs:element name="name" type="xs:string"/>
                <xs:element name="description"
type="xs:string"/>
        </xs:sequence>
</xs:complexType>
```

Step 14 - Quick introduction to different parts of a WSDL


WSDL
Endpoint
All Operations
Request Structure
Response Structure


element
types
complexType
message
message
definitions
input
portType
operation
output
input
binding
operation
output
service
port





Step 15 - Implementing SOAP Web Service for DeleteCourseDetailsRequest

/src/main/java/com/in28minutes/soap/webservices/soapcoursemanagement/soap/CourseDetailsEndpoint.java Modified

New Lines

```
@PayloadRoot(namespace = "http://in28minutes.com/courses",
localPart = "DeleteCourseDetailsRequest")
@ResponsePayload
public DeleteCourseDetailsResponse
deleteCourseDetailsRequest(
                @RequestPayload
```

```
 DeleteCourseDetailsRequest    request) {

        int status = service.deleteById(request.getId());

        DeleteCourseDetailsResponse response = new
DeleteCourseDetailsResponse();
        response.setStatus(status);

        return response;
}
```

/src/main/resources/course-details.xsd Modified

New Lines

```
        <xs:element name="DeleteCourseDetailsRequest">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name= "id"
type="xs:int"/>
                        </xs:sequence>
                </xs:complexType>

        </xs:element>

        <xs:element name="DeleteCourseDetailsResponse">
                <xs:complexType>
                        <xs:sequence>
                                <!-- 1 is success 0 for
failure -->
                                <xs:element name= "status"
type="xs:int"/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
```

Step 16 - Improving the DeleteCourseDetailsRequest - Using an Enum for Status

Step 17 - Exception Handling and SOAP Fault Responses

/src/main/java/com/in28minutes/soap/webservices/soapcoursemanagement/soap/CourseDetailsEndpoint.java Modified

```
        @PayloadRoot(namespace =
"http://in28minutes.com/courses", localPart =
"GetCourseDetailsRequest")
        @ResponsePayload
        public GetCourseDetailsResponse
processCourseDetailsRequest(@RequestPayload
GetCourseDetailsRequest request) {

                Course course =
service.findById(request.getId());

                if (course == null)
                        throw new
CourseNotFoundException("Invalid Course Id " +
request.getId());

                return mapCourseDetails(course);
        }


        @PayloadRoot(namespace =
"http://in28minutes.com/courses", localPart =
"DeleteCourseDetailsRequest")
        @ResponsePayload
        public DeleteCourseDetailsResponse
deleteCourseDetailsRequest(@RequestPayload
DeleteCourseDetailsRequest request) {

                Status status =
service.deleteById(request.getId());
```

```
                DeleteCourseDetailsResponse response = new
```

```
 DeleteCourseDetailsResponse();


        response.setStatus(mapStatus(status));

                return response;
        }

        private com.in28minutes.courses.Status
mapStatus(Status status) {
                if (status == Status.FAILURE)
                        return
com.in28minutes.courses.Status.FAILURE;
                return
com.in28minutes.courses.Status.SUCCESS;
        }
```

/src/main/java/com/in28minutes/soap/webservices/soapcoursemanagement/soap/exception/CourseNotFoundException.java New

```
package
com.in28minutes.soap.webservices.soapcoursemanagement.soap.
exception;

import
org.springframework.ws.soap.server.endpoint.annotation.Faul
tCode;
import

 org.springframework.ws.soap.server.endpoint.annotation.Soa
pFault;

@SoapFault(faultCode=FaultCode.CUSTOM,
        customFaultCode="
{http://in28minutes.com/courses}001_COURSE_NOT_FOUND")
public class CourseNotFoundException extends
RuntimeException {

        private static final long serialVersionUID =
```

```
3518170101751491969L;
```

```
        public CourseNotFoundException(String message) {
                super(message);


        }
    }
```

/src/main/java/com/in28minutes/soap/webservices/soapcoursemanagement/soap/service/CourseDetailsService.java Modified

```

        public enum Status {
                SUCCESS, FAILURE;
        }

        public Status deleteById(int id) {
                Iterator<Course> iterator =
courses.iterator();
                while (iterator.hasNext()) {
                        Course course = iterator.next();
                        if (course.getId() == id) {
                                iterator.remove();
                                return Status.SUCCESS;
                        }


        }
                return Status.FAILURE;
        }
```

/src/main/resources/course-details.xsd Modified

New Lines

```
        <xs:element name="DeleteCourseDetailsResponse">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name= "status"
```

```
 type="tns:Status"/>
                        </xs:sequence>
                </xs:complexType>

        </xs:element>

        <xs:simpleType    name="Status">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="SUCCESS"/>
                        <xs:enumeration value="FAILURE"/>
                </xs:restriction>
        </xs:simpleType>
```

Step 18 - Implementing Security for SOAP Web Services with WS Security

/example-files/Request-Security.xml New

```
<Envelope
xmlns="http://schemas.xmlsoap.org/soap/envelope/">
        <Header>
                <wsse:Security


        xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
1.0.xsd"
                        mustUnderstand="1">
                        <wsse:UsernameToken>

<wsse:Username>user</wsse:Username>

<wsse:Password>password</wsse:Password>
                        </wsse:UsernameToken>
                </wsse:Security>
        </Header>
```

```
        <Body>
                <GetCourseDetailsRequest
xmlns="http://in28minutes.com/courses">
                        <id>1</id>
                </GetCourseDetailsRequest>
        </Body>
</Envelope>
```

/example-files/Response-Fault.xml New

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Header />
        <SOAP-ENV:Body>
                <SOAP-ENV:Fault>
                        <faultcode
xmlns:ns0="http://in28minutes.com/courses">ns0:001_COURSE_N
OT_FOUND</faultcode>
                        <faultstring xml:lang="en">Invalid
Course Id 1234</faultstring>
                </SOAP-ENV:Fault>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

/pom.xml Modified

New Lines

```
        <dependency>

<groupId>org.springframework.ws</groupId>
                        <artifactId>spring-ws-
security</artifactId>
                        <exclusions>
                                <exclusion>
```

```
        <groupId>org.springframework.security</groupId>
                                        <artifactId>spring-
security-core</artifactId>
                                </exclusion>
                        </exclusions>
                </dependency>
                <dependency>
                        <groupId>com.sun.xml.wss</groupId>


        <artifactId>xws-security</artifactId>
                        <version>3.0</version>
                        <exclusions>
                                <exclusion>

<groupId>javax.xml.crypto</groupId>

<artifactId>xmldsig</artifactId>
                                </exclusion>
                        </exclusions>
                </dependency>
                <dependency>
                        <groupId>javax.activation</groupId>
                        <artifactId>activation</artifactId>
                        <version>1.1.1</version>
                </dependency>
```

/src/main/java/com/in28minutes/soap/webservices/soapcoursemanagement/soap/WebServiceConfig.java Modified

```



        //XwsSecurityInterceptor
        @Bean
        public XwsSecurityInterceptor
```

```
 securityInterceptor(){
                XwsSecurityInterceptor securityInterceptor
= new XwsSecurityInterceptor();
                //Callback Handler ->
SimplePasswordValidationCallbackHandler

securityInterceptor.setCallbackHandler(callbackHandler());
                //Security Policy -> securityPolicy.xml

securityInterceptor.setPolicyConfiguration(new
ClassPathResource("securityPolicy.xml"));
                return

 securityInterceptor;
        }

        @Bean
        public SimplePasswordValidationCallbackHandler
callbackHandler() {
                SimplePasswordValidationCallbackHandler
handler = new SimplePasswordValidationCallbackHandler();

handler.setUsersMap(Collections.singletonMap("user",
"password"));
                return handler;
        }

        //Interceptors.add -> XwsSecurityInterceptor
        @Override
        public void
addInterceptors(List<EndpointInterceptor> interceptors)

 {
                interceptors.add(securityInterceptor());
        }
```

/src/main/java/com/in28minutes/soap/webservices/soapcoursemanagement/soap/exception/CourseNotFoundException.java Modified

```
@SoapFault(faultCode = FaultCode.CUSTOM, customFaultCode =
"{http://in28minutes.com/courses}001_COURSE_NOT_FOUND")
public class CourseNotFoundException extends
RuntimeException {
```

/src/main/resources/course-details.xsd Modified

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://in28minutes.com/courses"
xmlns:tns="http://in28minutes.com/courses"


        elementFormDefault="qualified">

        <xs:element name="GetCourseDetailsRequest">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name="id"
type="xs:int" />
                        </xs:sequence>
                </xs:complexType>
        </xs:element>

        <xs:element name="GetCourseDetailsResponse">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element

 name="CourseDetails" type="tns:CourseDetails" />
                        </xs:sequence>
                </xs:complexType>
        </xs:element>

        <xs:element name="GetAllCourseDetailsRequest">
```

```
        <xs:complexType>
                </xs:complexType>
        </xs:element>

        <xs:element name="GetAllCourseDetailsResponse">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element
name="CourseDetails" type="tns:CourseDetails"

maxOccurs="unbounded" />
                        </xs:sequence>
                </xs:complexType>

        </xs:element>

        <xs:element name="DeleteCourseDetailsRequest">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name="id"
type="xs:int" />
                        </xs:sequence>
                </xs:complexType>
        </xs:element>

        <xs:element name="DeleteCourseDetailsResponse">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name="status"
type="tns:Status"

 />
                        </xs:sequence>
                </xs:complexType>
        </xs:element>

        <xs:simpleType
```

```
 name="Status">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="SUCCESS" />
                        <xs:enumeration value="FAILURE" />
                </xs:restriction>
        </xs:simpleType>

        <xs:complexType name="CourseDetails">
                <xs:sequence>
                        <xs:element name="id" type="xs:int"
/>
                        <xs:element name="name"
type="xs:string" />
                        <xs:element name="description"
type="xs:string" />
                </xs:sequence>

        </xs:complexType>
 </xs:schema>
```

/src/main/resources/securityPolicy.xml New

```
<?xml version="1.0" encoding="UTF-8"?>
<xwss:SecurityConfiguration
        xmlns:xwss="http://java.sun.com/xml/ns/xwss/config"
>
        <xwss:RequireUsernameToken
                passwordDigestRequired="false"
nonceRequired="false" />
</xwss:SecurityConfiguration>
```

Example Requests and Responses

```
<Envelope
xmlns="http://schemas.xmlsoap.org/soap/envelope/">
        <Header>
```

```
        <wsse:Security
                        xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
1.0.xsd"
                        mustUnderstand="1">
                        <wsse:UsernameToken>

<wsse:Username>user</wsse:Username>

<wsse:Password>password</wsse:Password>
                        </wsse:UsernameToken>
                </wsse:Security>
        </Header>
        <Body>
                <GetCourseDetailsRequest
xmlns="http://in28minutes.com/courses">
                        <id>1</id>
                </GetCourseDetailsRequest>
        </Body>


</Envelope>
```

/example-files/Request.xml

```
<Envelope
xmlns="http://schemas.xmlsoap.org/soap/envelope/">
        <Body>
                <GetCourseDetailsRequest
xmlns="http://in28minutes.com/courses">
                        <id>1</id>
                </GetCourseDetailsRequest>
        </Body>
</Envelope>
```

/example-files/Response-Fault.xml

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Header />
```

```
        <SOAP-ENV:Body>
                <SOAP-ENV:Fault>
                        <faultcode
xmlns:ns0="http://in28minutes.com/courses">ns0:001_COURSE_N
OT_FOUND</faultcode>
                        <faultstring xml:lang="en">Invalid
Course Id 1234</faultstring>
                </SOAP-ENV:Fault>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

/example-files/Response.xml

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <SOAP-ENV:Body>
        <ns2:GetCourseDetailsResponse
xmlns:ns2="http://in28minutes.com/courses">


 <ns2:CourseDetails>


 <ns2:id>1</ns2:id>
                 <ns2:name>Spring</ns2:name>
                <ns2:description>10 Steps</ns2:description>
              </ns2:CourseDetails>
         </ns2:GetCourseDetailsResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Restful Web Services with Spring Boot

Github Folder

https://github.com/in28minutes/spring-web-services/tree/master/restful-web-services

Restful Web Services with Spring Boot

- Step 01 - Initializing a RESTful Services Project with Spring Boot
- Step 02 - Understanding the RESTful Services we would create in this course
- Step 03 - Creating a Hello World Service
- Step 04 - Enhancing the Hello World Service to return a Bean
- Step 05 - Quick Review of Spring Boot Auto Configuration and Dispatcher Servlet
- Step 06 - Enhancing the Hello World Service with a Path Variable
- Step 07 - Creating User Bean and User Service
- Step 08 - Implementing GET Methods for User Resource
- Step 09 - Implementing POST Method to create User Resource
- Step 10 - Enhancing POST Method to return correct HTTP Status Code and Location

- Step 11 - Implementing Exception Handling - 404 Resource Not Found
- Step 12 - Implementing Generic Exception Handling for all Resources
- Step 13 - Exercise : User Post Resource and Exception Handling
- Step 14 - Implementing DELETE Method to delete a User Resource
- Step 15 - Implementing Validations for RESTful Services
- Step 16 - Implementing HATEOAS for RESTful Services
- Step 17 - Overview of Advanced RESTful Service Features
- Step 18 - Internationalization for RESTful Services
- Step 19 - Content Negotiation - Implementing Support for XML

- Step 20 - Configuring Auto Generation of Swagger Documentation

- Step 21 - Introduction to Swagger Documentation Format
- Step 22 - Enhancing Swagger Documentation with Custom Annotations
- Step 23 - Monitoring APIs with Spring Boot Actuator
- Step 24 - Implementing Static Filtering for RESTful Service
- Step 25 - Implementing Dynamic Filtering for RESTful Service
- Step 26 - Versioning RESTful Services - Basic Approach with URIs
- Step 27 - Versioning RESTful Services - Header and Content Negotiation Approach
- Step 28 - Implementing Basic Authentication with Spring Security
- Step 29 - Overview of Connecting RESTful Service to JPA
- Step 30 - Creating User Entity and some test data
- Step 31 - Updating GET methods on User Resource to use JPA
- Step 32 - Updating POST and DELETE methods on User Resource to use JPA
- Step 33 - Creating Post Entity and Many to One Relationship with User Entity
- Step 34 - Implementing a GET service to retrieve all Posts of a User
- Step 35 - Implementing a POST service to create a Post for a User
- Step 36 - Richardson Maturity Model
- Step 37 - RESTful Web Services - Best Practices

Useful Links

- POSTMAN - http://www.getpostman.com

Links from course examples

- Basic Resources
  - http://localhost:8080/hello-world
  - http://localhost:8080/hello-world-bean
  - http://localhost:8080/hello-world/path-variable/Ranga
  - http://localhost:8080/users/
  - http://localhost:8080/users/1

- JPA Resources
  - http://localhost:8080/jpa/users/
  - http://localhost:8080/jpa/users/1
  - http://localhost:8080/jpa/users/10001/posts

- Filtering

- - http://localhost:8080/filtering
  - http://localhost:8080/filtering-list

- Actuator
  - http://localhost:8080/actuator

- Versioning
  - http://localhost:8080/v1/person
  - http://localhost:8080/v2/person
  - http://localhost:8080/person/param

- - - params=[version=1]

- - http://localhost:8080/person/param

- - - params=[version=2]

  - http://localhost:8080/person/header

- - - headers=[X-API-VERSION=1]

  - http://localhost:8080/person/header

- - - headers=[X-API-VERSION=2]

- - http://localhost:8080/person/produces
    - produces=[application/vnd.company.app-v1+json]

  - http://localhost:8080/person/produces
    - produces=[application/vnd.company.app-v2+json]

- Swagger
  - http://localhost:8080/swagger-ui.html
  - http://localhost:8080/v2/api-docs

- H2-Console
  - http://localhost:8080/h2-console

Error in the Log

```
Resolved exception caused by Handler execution:
org.springframework.http.converter.HttpMessageNotWritableEx
ception:
No converter found for return value of type:
class
com.in28minutes.rest.webservices.restfulwebservices.HelloWo
rldBean
```

- This happened because there were no getters in HelloWorldBean class

Questions to Answer

- What is dispatcher servlet?

- Who is configuring dispatcher servlet?
- What does dispatcher servlet do?
- How does the HelloWorldBean object get converted to JSON?

- Who is configuring the error mapping?

- Mapping servlet: 'dispatcherServlet' to [/]

- Mapped "{[/hello-world],methods=[GET]}" onto public java.lang.String com.in28minutes.rest.webservices.restfulwebservices.HelloWorldController.helloWorld()

- Mapped "{[/hello-world-bean],methods=[GET]}" onto public com.in28minutes.rest.webservices.restfulwebservices.HelloWorldBean

- com.in28minutes.rest.webservices.restfulwebservices.HelloWorldController.helloWorldBean()
- Mapped "{[/error]}" onto public org.springframework.http.ResponseEntity<java.util.Map<java.lang.String, java.lang.Object>> org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.error(javax.servlet.http.HttpServletRequest)
- Mapped "{[/error],produces=[text/html]}" onto public org.springframework.web.servlet.ModelAndView

- org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.errorHtml(javax.servlet.http.HttpServletRequest,javax.servlet.http.HttpServletResponse)

Example Requests

GET http://localhost:8080/users

```
[
    {
        "id": 1,
        "name": "Adam",
        "birthDate":  "2017-07-19T04:40:20.796+0000"
    },


 {
        "id": 2,
        "name": "Eve",
        "birthDate": "2017-07-19T04:40:20.796+0000"



 },


 {
        "id": 3,
        "name":

 "Jack",
        "birthDate":

 "2017-07-19T04:40:20.796+0000"
    }
]
```

GET http://localhost:8080/users/1

```
{
    "id":
```

```
 1,
    "name": "Adam",
    "birthDate": "2017-07-19T04:40:20.796+0000"
}
```

POST http://localhost:8080/users

```
{
    "name": "Ranga",
    "birthDate": "2000-07-19T04:29:24.054+0000"
}
```

GET http://localhost:8080/users/1000

- Get request to a non existing resource.
- The response shows default error message structure auto configured by Spring Boot.

```
{
    "timestamp": "2017-07-19T05:28:37.534+0000",
    "status": 404,
    "error": "Not Found",
    "message":  "id-500",
    "path": "/users/500"

}
```

GET http://localhost:8080/users/1000

- Get request to a non existing resource.
- The response shows a Customized Message Structure

```
{
  "timestamp": "2017-07-19T05:31:01.961+0000",
  "message":  "id-500",
  "details": "Any details you would want to add"
```

-

```

```

-

```
}
```

POST http://localhost:8080/users with Validation Errors

Request

```
{
    "name": "R",
    "birthDate": "2000-07-19T04:29:24.054+0000"
}
```

Response - 400 Bad Request

```
{
    "timestamp":  "2017-07-19T09:00:27.912+0000",
    "message": "Validation Failed",
    "details":
"org.springframework.validation.BeanPropertyBindingResult:
1 errors\nField error in object 'user' on

 field 'name': rejected value [R]; codes
[Size.user.name,Size.name,Size.java.lang.String,Size];
arguments
[org.springframework.context.support.DefaultMessageSourceRe
solvable: codes [user.name,name]; arguments []; default
message  [name],2147483647,2]; default message [Name should
have atleast 2 characters]" }
```

GET http://localhost:8080/users/1 with HATEOAS

```
{
    "id": 1,
    "name":

 "Adam",
    "birthDate": "2017-07-19T09:26:18.337+0000",
    "_links":

 {
        "all-users": {
            "href":  "http://localhost:8080/users"


 }
    } }
```

XML Representation of Resources

GET http://localhost:8080/users

- Accept application/xml

```
<List>
     <item>
        <id>2</id>
        <name>Eve</name>
        <birthDate>2017-07-19T10:25:20.450+0000</birthDate>


 </item>
    <item>
        <id>3</id>
         <name>Jack</name>
         <birthDate>2017-07-
19T10:25:20.450+0000</birthDate>
      </item>
    <item>


 <id>4</id>
        <name>Ranga</name>
        <birthDate>2017-07-19T10:25:20.450+0000</birthDate>
    </item> </List>
```

POST http://localhost:8080/users

- Accept : application/xml
- Content-Type : application/xml

Request

```
<item>
          <name>Ranga</name>
        <birthDate>2017-07-19T10:25:20.450+0000</birthDate>
```

```
 </item>
```

Response

- Status - 201 Created

Generating Swagger Documentation

```
  public static final Contact DEFAULT_CONTACT = new
Contact(
      "Ranga Karanam", "http://www.in28minutes.com",

 "in28minutes@gmail.com");

  public static final ApiInfo DEFAULT_API_INFO = new

 ApiInfo(
      "Awesome API Title", "Awesome API Description",
"1.0",
      "urn:tos",  DEFAULT_CONTACT,
      "Apache 2.0",
"http://www.apache.org/licenses/LICENSE-2.0");

  private static final Set<String>

 DEFAULT_PRODUCES_AND_CONSUMES =
      new HashSet<String>(Arrays.asList("application/json",
           "application/xml"));

  @Bean
  public Docket api() {
    return new

 Docket(DocumentationType.SWAGGER_2)
         .apiInfo(DEFAULT_API_INFO)
        .produces(DEFAULT_PRODUCES_AND_CONSUMES)
         .consumes(DEFAULT_PRODUCES_AND_CONSUMES);
  }
```

Resource Method description

```
   @GetMapping("/users/{id}")
  @ApiOperation(value = "Finds Users by

 id",
    notes = "Also returns a link to retrieve all users with
rel -  all-users")
  public Resource<User> retrieveUser(@PathVariable int id)
{
```

API Model

```
@ApiModel(value="User Details", description="Contains all
details of a user")
public class User


 {

  @Size(min=2, message="Name should have atleast 2
characters")
  @ApiModelProperty(notes = "Name should have atleast 2
characters")
  private String

 name;

  @Past
  @ApiModelProperty(notes = "Birth Date should be in the
Past")
  private Date birthDate;
```

Filtering

Code

```
@JsonIgnoreProperties(value={"field1"}) public class
SomeBean {

  private String
```

```
 field1;

  @JsonIgnore
  private String field2;

  private String field3;
```

Response

```
{
    "field3": "value3"
}
```

Versioning

- Media type versioning (a.k.a “content negotiation” or “accept header”)
  - GitHub
- (Custom) headers versioning
  - Microsoft
- URI Versioning
  - Twitter
- Request Parameter versioning
  - Amazon
- Factors
- URI Pollution
- Misuse of HTTP Headers
- Caching
- Can we execute the request on the browser?
- API Documentation
- No Perfect Solution

More

- https://www.mnot.net/blog/2011/10/25/web_api_versioning_smackdown
- http://urthen.github.io/2013/05/09/ways-to-version-your-api/

- http://stackoverflow.com/questions/389169/best-practices-for-api-versioning
- http://www.lexicalscope.com/blog/2012/03/12/how-are-rest-apis-versioned/
- https://www.3scale.net/2016/06/api-versioning-methods-a-brief-reference/

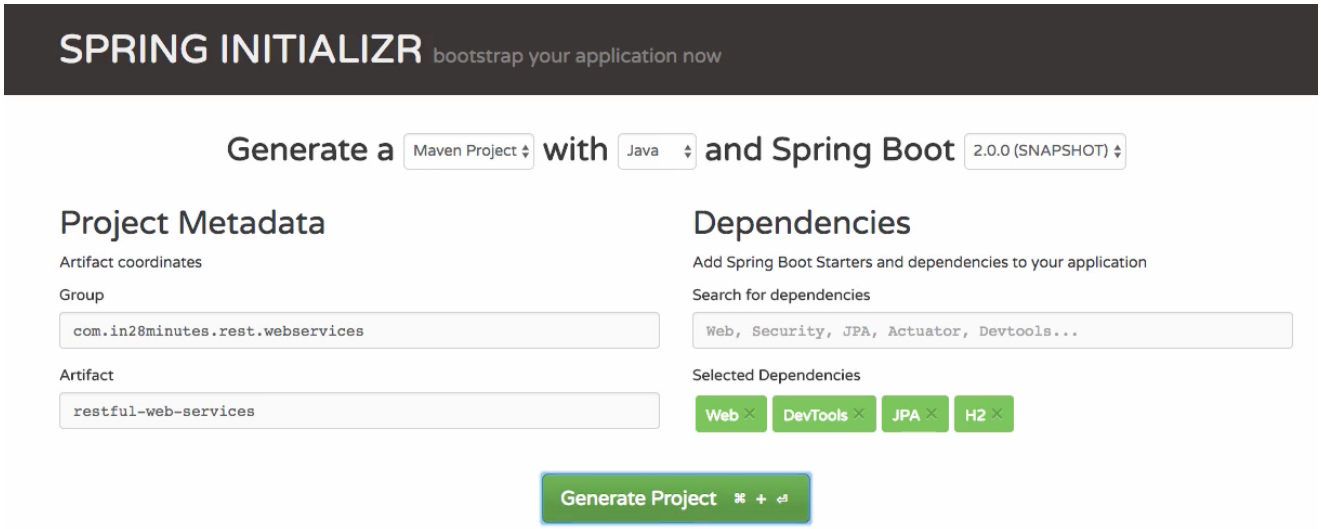
Table Structure

```
create table user (
id integer not null,  birth_date timestamp,

name varchar(255),

primary key (id)
);

create table post (
id integer not null,
description varchar(255),
user_id integer,
primary key (id) );


alter table post  add constraint post_to_user_foreign_key

foreign key (user_id) references user;
```

Step 01 - Initializing a RESTful Services Project with Spring Boot

Creating a Spring Project with Spring Initializr is a cake walk.

> *Spring Initializr http://start.spring.io/ is great tool to bootstrap your Spring Boot projects.*



As shown in the image above, following steps have to be done

- Launch Spring Initializr and choose the following
-
  - Choose `com.in28minutes.rest.webservices` as Group
  - Choose `restful-web-services` as Artifact
  - Choose Release >= 2.0.0 (Avoid SNAPSHOT!)
  - Choose following dependencies
    - Web
    - DevTools
    - JPA
    - H2
- Click Generate Project.
- Import the project into Eclipse.

Step 02 - Understanding the RESTful Services we would create

Social Media Application Resource Mappings
User -> Posts

- Retrieve all Users - GET /users
- Create a User - POST /users
- Retrieve one User - GET /users/{id} -> /users/1
- Delete a User - DELETE /users/{id} -> /users/1

- Retrieve all posts for a User - GET /users/{id}/posts
- Create a posts for a User - POST /users/{id}/posts
- Retrieve details of a post - GET /users/{id}/posts/{post_id}

Step 03 - Creating a Hello World Service

```
@RestController
public class HelloWorldController    {

  @GetMapping(path =

 "/hello-world")
  public String helloWorld() {
    return "Hello World";
  }
}
```

Step 04 - Enhancing the Hello World Service to return a Bean

```
 @GetMapping(path =   "/hello-world-bean")
  public HelloWorldBean helloWorldBean() {
    return new HelloWorldBean("Hello  World");
  }
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/HelloWorldBean.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices;

public class HelloWorldBean  {

  private String
```

```
 message;

  public HelloWorldBean(String message)  {
    this.message =

 message;
  }

  public String getMessage()  {
    return message;
  }

  public void setMessage(String message)   {
    this.message = message;


 }

   @Override
  public String toString() {
    return String.format("HelloWorldBean [message=%s]",
message);
  }

}
```

Step 05 - Quick Review of Spring Boot Auto Configuration and Dispatcher Servlet

Let us understand Spring Boot Auto Configuration in depth

- http://www.springboottutorial.com/spring-boot-auto-configuration

Step 06 - Enhancing the Hello World Service with a Path Variable

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/HelloWorldController.java

```
package
com.in28minutes.rest.webservices.restfulwebservices;

import org.springframework.web.bind.annotation.GetMapping;
import

 org.springframework.web.bind.annotation.PathVariable;
import
org.springframework.web.bind.annotation.RestController;

//Controller @RestController public class
HelloWorldController {

  @GetMapping(path =

 "/hello-world")
  public String helloWorld() {
    return "Hello World";
   }

  @GetMapping(path = "/hello-world-bean")
  public HelloWorldBean helloWorldBean()

 {
    return new HelloWorldBean("Hello World");
  }

  ///hello-world/path-variable/in28minutes
  @GetMapping(path = "/hello-world/path-variable/{name}")
  public HelloWorldBean
helloWorldPathVariable(@PathVariable String name)

 {
    return new HelloWorldBean(String.format("Hello World,
%s",
```

```
 name));
  }


}
```

/src/main/resources/application.properties Modified

New Lines

```
logging.level.org.springframework = info
```

Step 07 - Creating User Bean and User Service

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/HelloWorldBean.java

Package Change

```
package
com.in28minutes.rest.webservices.restfulwebservices.helloworld;
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/HelloWorldController.java

Package Change

```
package
com.in28minutes.rest.webservices.restfulwebservices.helloworld;
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/User.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.user;

import java.util.Date;
 public class User
```

```
{

 private Integer   id;

 private String

name;

 private Date

birthDate;

 public User(Integer id, String name, Date birthDate) {
   super();
   this.id = id;
   this.name = name;
   this.birthDate = birthDate;


}

 public Integer getId()   {
   return  id;
  }

 public void setId(Integer id) {
   this.id =

id;
 }

 public String getName() {
   return name;
 }

 public void setName(String name)  {
   this.name =
```

```
 name;
  }

  public Date getBirthDate() {
    return  birthDate;
   }

  public void setBirthDate(Date birthDate) {
    this.birthDate =  birthDate;


 }



 @Override
  public String toString() {
    return String.format("User [id=%s, name=%s,
birthDate=%s]", id, name, birthDate);
  }
 }
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserDaoService.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.user;

import java.util.ArrayList;
import java.util.Date;

import java.util.List;

import org.springframework.stereotype.Component;

@Component public class UserDaoService

 {
  private static List<User> users = new
```

```
ArrayList<>();

 private static int usersCount = 3;

 static {
   users.add(new User(1, "Adam", new Date()));
   users.add(new User(2, "Eve", new  Date()));
   users.add(new User(3, "Jack", new  Date()));
 }

 public List<User> findAll()

{
   return users;
  }

 public User save(User user)

{
   if (user.getId() == null) {
      user.setId(++usersCount);
    }
   users.add(user);
   return   user;


}

 public User findOne(int id) {
   for (User user : users) {
     if (user.getId() == id) {
       return user;



}
```

```
 }
    return null;
  }

}
```

Step 08 - Implementing GET Methods for User Resource

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserResource.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.user;

import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.PathVariable;
import
org.springframework.web.bind.annotation.RestController;

@RestController public class UserResource

 {

   @Autowired
  private UserDaoService

 service;

  @GetMapping("/users")
  public List<User> retrieveAllUsers() {
    return  service.findAll();
```

```
 }

  @GetMapping("/users/{id}")
  public User retrieveUser(@PathVariable int id) {
    return service.findOne(id);
  }

}
```

/src/main/resources/application.properties Modified

New Lines

```
#This is not really needed as this is the default after
2.0.0.RELEASE spring.jackson.serialization.write-dates-as-
timestamps=false
```

Step 09 - Implementing POST Method to create User Resource

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserResource.java Modified

```
   // input - details of user
  // output - CREATED & Return the created URI


 @PostMapping("/users")
  public void createUser(@RequestBody User    user){
    User savedUser = service.save(user);
  }
```

Step 10 - Enhancing POST Method to return correct HTTP Status Code and Location

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserResource.java Modified

```
```

```
 // input - details of  user
  // output - CREATED & Return the created


 URI


 @PostMapping("/users")
  public ResponseEntity<Object> createUser(@RequestBody
User user) {
    User savedUser = service.save(user);
    // CREATED
    // /user/{id}     savedUser.getId()

    URI location = ServletUriComponentsBuilder
      .fromCurrentRequest()
       .path("/{id}")


 .buildAndExpand(savedUser.getId()).toUri();

    return   ResponseEntity.created(location).build();

  }
```

Step 11 - Implementing Exception Handling - 404 Resource Not Found

Step 12 - Implementing Generic Exception Handling for all Resources

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/exception/CustomizedResponseEntityExceptionHandler.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.excepti
on;

import java.util.Date;
```

```
 import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

import

 org.springframework.web.bind.annotation.ControllerAdvice;

import
org.springframework.web.bind.annotation.ExceptionHandler;

import
org.springframework.web.bind.annotation.RestController;
import org.springframework.web.context.request.WebRequest;
import
org.springframework.web.servlet.mvc.method.annotation.Respo
nseEntityExceptionHandler;
 import
com.in28minutes.rest.webservices.restfulwebservices.user.Us
erNotFoundException;

@ControllerAdvice @RestController
public class CustomizedResponseEntityExceptionHandler
extends ResponseEntityExceptionHandler  {

  @ExceptionHandler(Exception.class)
  public final ResponseEntity<Object>

 handleAllExceptions(Exception ex, WebRequest request)  {
    ErrorDetails errorDetails = new ErrorDetails(new
Date(),  ex.getMessage(),
        request.getDescription(false));
    return new ResponseEntity(errorDetails,
HttpStatus.INTERNAL_SERVER_ERROR);
   }

    @ExceptionHandler(UserNotFoundException.class)
  public final ResponseEntity<Object>
handleUserNotFoundException(UserNotFoundException ex,
WebRequest request) {
    ErrorDetails errorDetails = new ErrorDetails(new
```

```
 Date(), ex.getMessage(),


 request.getDescription(false));
    return new ResponseEntity(errorDetails,

 HttpStatus.NOT_FOUND);
  }
 }
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/exception/ErrorDetails.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.excepti
on;

import java.util.Date;

public class ErrorDetails {
  private Date  timestamp;
  private String message;
  private String  details;

  public ErrorDetails(Date timestamp, String message,

 String details) {
     super();
    this.timestamp =  timestamp;
    this.message = message;
    this.details = details;
   }

  public Date getTimestamp()  {
    return timestamp;
   }

  public String getMessage()  {
    return
```

```
 message;


 }

  public String getDetails()


 {
    return details;
  }

}
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserNotFoundException.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.user;

import org.springframework.http.HttpStatus; import
org.springframework.web.bind.annotation.ResponseStatus;
 @ResponseStatus(HttpStatus.NOT_FOUND)
public class UserNotFoundException extends RuntimeException
{
  public UserNotFoundException(String message) {


 super(message);
  }
}
```

Step 13 - Exercise : User Post Resource and Exception Handling

Step 14 - Implementing DELETE Method to delete a User Resource

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserDaoService.java Modified

```
public User deleteById(int id) {
   Iterator<User> iterator = users.iterator();
   while (iterator.hasNext()) {
     User user =

iterator.next();
     if (user.getId() == id)

{
       iterator.remove();
       return user;
     }



}
   return null;
 }
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserResource.java Modified

```
 @GetMapping("/users/{id}")
 public User retrieveUser(@PathVariable int id) {
   User user =   service.findOne(id);



if(user==null)
     throw new UserNotFoundException("id-"+ id);

   return  user;
 }

 @DeleteMapping("/users/{id}")
 public void deleteUser(@PathVariable int id)  {
   User user =
```

```
service.deleteById(id);

   if(user==null)
     throw new UserNotFoundException("id-"+  id);
 }

 //
 // input - details of user
 // output - CREATED & Return the created

URI
 @PostMapping("/users")
 public ResponseEntity<Object> createUser(@RequestBody

User user)

{
   User savedUser = service.save(user);
   // CREATED
   // /user/{id}

savedUser.getId()

   URI location = ServletUriComponentsBuilder


.fromCurrentRequest()
     .path("/{id}")


.buildAndExpand(savedUser.getId()).toUri();

   return  ResponseEntity.created(location).build();

 }
```

Step 15 - Implementing Validations for RESTful Services

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/exception/CustomizedResponseEntityExceptionHandler.java Modified

```
@Override
protected ResponseEntity<Object>
handleMethodArgumentNotValid(MethodArgumentNotValidExceptio
n ex,
    HttpHeaders headers, HttpStatus status, WebRequest
request) {
  ErrorDetails errorDetails = new ErrorDetails(new Date(),
"Validation  Failed",
      ex.getBindingResult().toString());
  return new ResponseEntity(errorDetails,

 HttpStatus.BAD_REQUEST);
}
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/User.java Modified

```
 @Size(min=2, message="Name should have atleast 2
characters")
  private String

 name;



 @Past
  private Date birthDate;
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserResource.java Modified

```
public ResponseEntity<Object> createUser(@Valid
@RequestBody User user) {
```

Step 16 - Implementing HATEOAS for RESTful Services

/pom.xml Modified

```
     <dependency>
       <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-hateoas</artifactId>
    </dependency>
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserResource.java Modified

```
   @GetMapping("/users/{id}")
  public Resource<User> retrieveUser(@PathVariable int id)

 {
    User user =  service.findOne(id);

    if(user==null)
      throw new UserNotFoundException("id-"+ id);



    //"all-users", SERVER_PATH +

 "/users"
     //retrieveAllUsers
    Resource<User> resource = new Resource<User>(user);

    ControllerLinkBuilder linkTo  =


 linkTo(methodOn(this.getClass()).retrieveAllUsers());

    resource.add(linkTo.withRel("all-users"));

    //HATEOAS

    return resource;
```

```
 }


   //HATEOAS

   @PostMapping("/users")
  public ResponseEntity<Object> createUser(@Valid
@RequestBody User user) {
    User savedUser = service.save(user);
    // CREATED
    // /user/{id}       savedUser.getId()

    URI location = ServletUriComponentsBuilder


 .fromCurrentRequest()
      .path("/{id}")
      .buildAndExpand(savedUser.getId()).toUri();

    return ResponseEntity.created(location).build();

  }
```

Step 17 - Overview of Advanced RESTful Service Features

- Step 18 - Internationalization for RESTful Services
- Step 19 - Content Negotiation - Implementing Support for XML
- Step 20 - Configuring Auto Generation of Swagger Documentation
- Step 21 - Introduction to Swagger Documentation Format

- Step 22 - Enhancing Swagger Documentation with Custom Annotations

- Step 23 - Monitoring APIs with Spring Boot Actuator
- Step 24 - Implementing Static Filtering for RESTful Service
- Step 25 - Implementing Dynamic Filtering for RESTful Service
- Step 26 - Versioning RESTful Services - Basic Approach with URIs

- Step 27 - Versioning RESTful Services - Header and Content Negotiation Approach
- Step 28 - Implementing Basic Authentication with Spring Security

Step 18 - Internationalization for RESTful Services

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/RestfulWebServicesApplication.java Modified

New Lines

```
import java.util.Locale;
import org.springframework.context.annotation.Bean;
import
org.springframework.context.support.ResourceBundleMessageSo
urce; import
org.springframework.web.servlet.LocaleResolver; import

 org.springframework.web.servlet.i18n.SessionLocaleResolver
;

  @Bean
  public LocaleResolver localeResolver() {
    SessionLocaleResolver localeResolver = new
SessionLocaleResolver();
     localeResolver.setDefaultLocale(Locale.US);
    return  localeResolver;


 }



 @Bean
  public ResourceBundleMessageSource messageSource() {
    ResourceBundleMessageSource messageSource = new
ResourceBundleMessageSource();
     messageSource.setBasename("messages");
    return
```

```
 messageSource;
  }
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/helloworld/HelloWorldController.java Modified

New Lines

```
import java.util.Locale;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource; import
org.springframework.web.bind.annotation.RequestHeader;

@Autowired
private MessageSource messageSource;
 @GetMapping(path = "/hello-world-internationalized")
public String helloWorldInternationalized(
    @RequestHeader(name="Accept-Language", required=false)
Locale locale) {
  return messageSource.getMessage("good.morning.message",
null, locale); }
```

/src/main/resources/messages.properties New

```
good.morning.message=Good Morning
```

/src/main/resources/messages_fr.properties New

```
good.morning.message=Bonjour
```

/src/main/resources/messages_nl.properties New

```
good.morning.message=Goede Morgen
```

Step 18 Part 2 - Simplifying Internationalization for RESTful Services

Use AcceptHeaderLocaleResolver

```
@SpringBootApplication
public class RestfulWebServicesApplication   {

  ....

  @Bean
  public LocaleResolver localeResolver()  {
    AcceptHeaderLocaleResolver localeResolver = new
AcceptHeaderLocaleResolver();
      localeResolver.setDefaultLocale(Locale.US);
    return localeResolver;
  }
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/helloworld/HelloWorldController.java

```
@GetMapping(path = "/hello-world-internationalized")
  public String helloWorldInternationalized() {
    return messageSource.getMessage("good.morning.message",
null,
                  LocaleContextHolder.getLocale());
  }
```

Use MessageSource configuration from application.properties

```
spring.messages.basename=messages
```

Step 19 - Content Negotiation - Implementing Support for XML

/pom.xml Modified

New Lines

```
 <dependency>
```

```
 <groupId>com.fasterxml.jackson.dataformat</groupId>
      <artifactId>jackson-dataformat-xml</artifactId>
    </dependency>
```

Step 20 - Configuring Auto Generation of Swagger Documentation

Step 21 - Introduction to Swagger Documentation Format

Step 22 - Enhancing Swagger Documentation with Custom Annotations

/pom.xml Modified

New Lines

```
    <dependency>
       <groupId>io.springfox</groupId>
      <artifactId>springfox-swagger2</artifactId>
      <version>2.4.0</version>
     </dependency>

    <dependency>
      <groupId>io.springfox</groupId>
      <artifactId>springfox-swagger-ui</artifactId>
      <version>2.4.0</version>
    </dependency>
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/SwaggerConfig.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices;

import java.util.Arrays; import java.util.HashSet;
import java.util.Set;
```

```
import org.springframework.context.annotation.Bean;
import
org.springframework.context.annotation.Configuration;

import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import
springfox.documentation.swagger2.annotations.EnableSwagger2
;
 @Configuration
@EnableSwagger2 public class SwaggerConfig {

  public static final Contact DEFAULT_CONTACT = new

 Contact(
      "Ranga Karanam", "http://www.in28minutes.com",
"in28minutes@gmail.com");

  public static final ApiInfo DEFAULT_API_INFO = new
ApiInfo(
      "Awesome API Title", "Awesome API Description",
"1.0",
      "urn:tos", DEFAULT_CONTACT,
      "Apache 2.0",
"http://www.apache.org/licenses/LICENSE-2.0");

  private static final Set<String>

 DEFAULT_PRODUCES_AND_CONSUMES =
      new HashSet<String>(Arrays.asList("application/json",
           "application/xml"));



 @Bean
  public Docket api()
```

```
 {
    return new Docket(DocumentationType.SWAGGER_2)
         .apiInfo(DEFAULT_API_INFO)
        .produces(DEFAULT_PRODUCES_AND_CONSUMES)
        .consumes(DEFAULT_PRODUCES_AND_CONSUMES);
  }

}
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/UserApiDocumentationConfig.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices;

import io.swagger.annotations.Contact; import
io.swagger.annotations.ExternalDocs;

import io.swagger.annotations.Info;
import io.swagger.annotations.License;
import io.swagger.annotations.SwaggerDefinition;
  @SwaggerDefinition(
        info = @Info(
                description = "Awesome Resources",
                version = "V12.0.12",
                title = "Awesome Resource API",
                contact = @Contact(
                   name = "Ranga Karanam",
                   email =

 "ranga.karanam@in28minutes.com",
                   url = "http://www.in28minutes.com"
                ),
                license =

 @License(
                   name = "Apache 2.0",
                   url =
```

```
 "http://www.apache.org/licenses/LICENSE-2.0"
                 )
        ),
        consumes = {"application/json", "application/xml"},
        produces = {"application/json", "application/xml"},
        schemes = {SwaggerDefinition.Scheme.HTTP,
SwaggerDefinition.Scheme.HTTPS},
        externalDocs = @ExternalDocs(value = "Read This For
Sure", url = "http://in28minutes.com")

) public interface UserApiDocumentationConfig {

}
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/User.java Modified

```
@ApiModel(description="All details about the user. ")
public class User  {

  private Integer  id;

  @Size(min=2, message="Name should have atleast 2
characters")
  @ApiModelProperty(notes="Name should have atleast 2
characters")
  private String

 name;

  @Past
  @ApiModelProperty(notes="Birth date should be in the
past")
  private Date birthDate;
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserResource.java Modified

```
 @GetMapping("/users/{id}")
  public Resource<User> retrieveUser(@PathVariable int id)
{
    User user =  service.findOne(id);

    if(user==null)
      throw new UserNotFoundException("id-"+   id);


    //"all-users", SERVER_PATH +

 "/users"
    //retrieveAllUsers
    Resource<User> resource = new  Resource<User>(user);

    ControllerLinkBuilder linkTo

 =

linkTo(methodOn(this.getClass()).retrieveAllUsers());

     resource.add(linkTo.withRel("all-users"));



 //HATEOAS

    return resource;
  }
```

Step 23 - Monitoring APIs with Spring Boot Actuator

/pom.xml Modified

New Lines

```
```

```
 <dependency>


 <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-actuator</artifactId>
     </dependency>

     <dependency>
      <groupId>org.springframework.data</groupId>
       <artifactId>spring-data-rest-hal-
browser</artifactId>
    </dependency>
```

application.properties Modified

```
management.endpoints.web.exposure.include=*
```

Step 24 - Implementing Static Filtering for RESTful Service

Step 25 - Implementing Dynamic Filtering for RESTful Service

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/UserApiDocumentationConfig.java Deleted

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/filtering/FilteringController.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.filteri
ng;

import java.util.Arrays;
import java.util.List;

import
org.springframework.http.converter.json.MappingJacksonValue
;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RestController;
 import com.fasterxml.jackson.databind.ser.FilterProvider;
```

```
import

 com.fasterxml.jackson.databind.ser.impl.SimpleBeanProperty
Filter;

import
com.fasterxml.jackson.databind.ser.impl.SimpleFilterProvide
r;

@RestController public class FilteringController {

  //  field1,field2
  @GetMapping("/filtering")
  public MappingJacksonValue retrieveSomeBean()  {
    SomeBean someBean = new SomeBean("value1", "value2",
"value3");

    SimpleBeanPropertyFilter filter =
SimpleBeanPropertyFilter.filterOutAllExcept("field1",

 "field2");

    FilterProvider filters = new
SimpleFilterProvider().addFilter("SomeBeanFilter",

 filter);

    MappingJacksonValue mapping = new
MappingJacksonValue(someBean);

    mapping.setFilters(filters);

    return mapping;
  }

  // field2, field3
   @GetMapping("/filtering-list")
  public MappingJacksonValue retrieveListOfSomeBeans()
```

```
 {
    List<SomeBean> list = Arrays.asList(new
SomeBean("value1", "value2",  "value3"),
        new SomeBean("value12", "value22",

 "value32"));

    SimpleBeanPropertyFilter filter =
SimpleBeanPropertyFilter.filterOutAllExcept("field2",

 "field3");

    FilterProvider filters = new
SimpleFilterProvider().addFilter("SomeBeanFilter",
filter);

    MappingJacksonValue mapping = new
MappingJacksonValue(list);

    mapping.setFilters(filters);

    return mapping;
  }

}
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/filtering/SomeBean.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.filteri
ng;

import com.fasterxml.jackson.annotation.JsonFilter;
  @JsonFilter("SomeBeanFilter")
public class SomeBean {

  private String  field1;

  private String
```

```
field2;

 private String  field3;

 public SomeBean(String field1, String field2, String

field3) {
   super();
   this.field1 = field1;
   this.field2 = field2;
   this.field3 =

field3;
  }

 public String getField1() {
   return field1;
  }

 public void setField1(String field1) {
   this.field1 = field1;
 }

 public String getField2()

{
   return field2;


}

 public void setField2(String field2) {
   this.field2 = field2;
  }

 public String getField3()   {
   return
```

```
 field3;
   }

  public void setField3(String field3) {
    this.field3 = field3;
  }
 }
```

Step 26 - Versioning RESTful Services - Basic Approach with URIs

Step 27 - Versioning RESTful Services - Header and Content Negotiation Approach

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/versioning/Name.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.version
ing;

public class Name  {
  private String firstName;
  private String lastName;

  public Name()

 {
  }

  public Name(String firstName, String lastName) {




 super();
    this.firstName = firstName;
    this.lastName = lastName;
```

```
 }

  public String getFirstName() {
    return firstName;
  }

  public void setFirstName(String firstName)  {
    this.firstName = firstName;


 }

  public String getLastName() {
    return lastName;
   }

  public void setLastName(String lastName) {
    this.lastName = lastName;
  }

}
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/versioning/PersonV1.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.version
ing;
 public class PersonV1


 {
  private String name;

  public PersonV1() {
     super();
  }

  public PersonV1(String name)
```

```
 {


 super();
    this.name = name;
   }

  public String getName()  {
    return name;


 }

  public void setName(String name) {
    this.name = name;
  }



}
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/versioning/PersonV2.java New

```
package

 com.in28minutes.rest.webservices.restfulwebservices.versio
ning;

public class PersonV2

 {
  private Name  name;

  public PersonV2() {
    super();
  }

  public PersonV2(Name name)
```

```
 {
    super();
    this.name = name;
  }

  public Name getName()


 {
    return   name;


 }

  public void setName(Name name) {
    this.name = name;
  }

}
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/versioning/PersonVersioningController.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.version
ing;
 import org.springframework.web.bind.annotation.GetMapping;
import

 org.springframework.web.bind.annotation.RestController;

@RestController

public class PersonVersioningController  {

  @GetMapping("v1/person")
  public PersonV1 personV1() {
    return new PersonV1("Bob Charlie");
```

```
 }

  @GetMapping("v2/person")
  public PersonV2 personV2() {
    return new PersonV2(new Name("Bob", "Charlie"));
   }

  @GetMapping(value = "/person/param", params =

 "version=1")
  public PersonV1 paramV1()   {
    return new PersonV1("Bob Charlie");


 }

  @GetMapping(value = "/person/param", params =
"version=2")
  public PersonV2 paramV2() {
    return new PersonV2(new Name("Bob", "Charlie"));
  }

  @GetMapping(value = "/person/header", headers = "X-API-
VERSION=1")
  public PersonV1 headerV1() {
    return new PersonV1("Bob  Charlie");
  }

  @GetMapping(value = "/person/header", headers =

 "X-API-VERSION=2")
  public PersonV2 headerV2()

 {
    return new PersonV2(new Name("Bob", "Charlie"));
```

```
 }

  @GetMapping(value = "/person/produces", produces =
"application/vnd.company.app-v1+json")
  public PersonV1 producesV1() {
    return new PersonV1("Bob Charlie");
  }

  @GetMapping(value = "/person/produces", produces =
"application/vnd.company.app-v2+json")
  public PersonV2 producesV2()   {
    return new PersonV2(new Name("Bob",  "Charlie"));
  }

}
```

Step 28 - Implementing Basic Authentication with Spring Security

/pom.xml Modified

New Lines

```
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
```

/src/main/resources/application.properties Modified

New Lines

```
spring.security.filter.dispatcher-types=request

spring.security.user.name=username

spring.security.user.password=password
```

Step 29 - Overview of Connecting RESTful Service to JPA

- Step 30 - Creating User Entity and some test data
- Step 31 - Updating GET methods on User Resource to use JPA
- Step 32 - Updating POST and DELETE methods on User Resource to use JPA
- Step 33 - Creating Post Entity and Many to One Relationship with User Entity
- Step 34 - Implementing a GET service to retrieve all Posts of a User
- Step 35 - Implementing a POST service to create a Post for a User

Step 30 - Creating User Entity and some test data

Step 31 - Updating GET methods on User Resource to use JPA

Step 32 - Updating POST and DELETE methods on User Resource to use JPA

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/User.java Modified

New Lines

```
@ApiModel(description="All details about the user. ")

@Entity
public class User {

  @Id
  @GeneratedValue
  private Integer id;
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserJPAResource.java New

```
package

 com.in28minutes.rest.webservices.restfulwebservices.user;

import static
```

```
 org.springframework.hateoas.mvc.ControllerLinkBuilder.link
To;
import static
org.springframework.hateoas.mvc.ControllerLinkBuilder.metho
dOn;

import java.net.URI;
import java.util.List;
import java.util.Optional;

import javax.validation.Valid;
  import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.hateoas.Resource; import
org.springframework.hateoas.mvc.ControllerLinkBuilder;
import org.springframework.http.ResponseEntity; 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.RequestBody;

import
org.springframework.web.bind.annotation.RestController;
import
org.springframework.web.servlet.support.ServletUriComponent
sBuilder;

@RestController
public class UserJPAResource   {



 @Autowired
  private UserDaoService
```

```
service;



@Autowired
 private UserRepository userRepository;

 @GetMapping("/jpa/users")
 public List<User> retrieveAllUsers() {
   return  userRepository.findAll();
  }

 @GetMapping("/jpa/users/{id}")
 public Resource<User> retrieveUser(@PathVariable int id)
{
   Optional<User> user =  userRepository.findById(id);

   if(!user.isPresent())
     throw new UserNotFoundException("id-"+  id);

   //"all-users", SERVER_PATH + "/users"


//retrieveAllUsers
   Resource<User> resource = new

Resource<User>(user.get());

   ControllerLinkBuilder linkTo =


linkTo(methodOn(this.getClass()).retrieveAllUsers());



resource.add(linkTo.withRel("all-users"));
```

```
 //HATEOAS

    return resource;
  }

  @DeleteMapping("/jpa/users/{id}")
  public void deleteUser(@PathVariable int id)

 {
    User user =  service.deleteById(id);

    if(user==null)
      throw new UserNotFoundException("id-"+  id);
   }

  //
  // input - details of user
  // output - CREATED & Return the created URI

  //HATEOAS

  @PostMapping("/jpa/users")
  public ResponseEntity<Object> createUser(@Valid
@RequestBody User user) {
    User savedUser =  service.save(user);

    URI location =

 ServletUriComponentsBuilder
      .fromCurrentRequest()
      .path("/{id}")


 .buildAndExpand(savedUser.getId()).toUri();

    return
```

```
 ResponseEntity.created(location).build();

   }

}
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserRepository.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.user;
 import

 org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
 @Repository public interface UserRepository extends
JpaRepository<User, Integer>{

}
```

/src/main/resources/application.properties Modified

New Lines

```
management.endpoints.web.exposure.include=*
spring.jpa.show-sql=true
spring.h2.console.enabled=true
```

/src/main/resources/data.sql New

```
insert into user values(1, sysdate(), 'AB'); insert into
user values(2, sysdate(), 'Jill');

insert into user values(3, sysdate(), 'Jam');
```

Step 33 - Creating Post Entity and Many to One Relationship with User Entity

Step 34 - Implementing a GET service to retrieve all Posts of a

User

Step 35 - Implementing a POST service to create a Post for a User

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/Post.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.user;

import javax.persistence.Entity;
import javax.persistence.FetchType; import
javax.persistence.GeneratedValue; import
javax.persistence.Id;
import javax.persistence.ManyToOne;
 import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
public class Post {

  @Id
  @GeneratedValue
  private Integer id;
  private String description;

   @ManyToOne(fetch=FetchType.LAZY)


 @JsonIgnore
  private User


 user;

  public Integer getId()  {
    return  id;
```

```
 }

  public void setId(Integer id) {
    this.id = id;


 }

  public String getDescription() {
    return description;
   }

  public void setDescription(String description)

 {
    this.description = description;
  }

  public User getUser() {
    return user;
  }

  public void setUser(User user) {
    this.user = user;
  }

   @Override
  public String toString() {
    return String.format("Post [id=%s, description=%s]",
id,

 description);
  }
   }
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/PostRepository.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.user;

import
org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
 @Repository

public interface PostRepository extends JpaRepository<Post,
Integer>{
 }
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/User.java Modified

```
   @OneToMany(mappedBy="user")
  private List<Post> posts;
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserJPAResource.java Modified

```
@RestController
public class UserJPAResource  {

  @Autowired
  private UserRepository

 userRepository;

   @Autowired
  private PostRepository  postRepository;

  @GetMapping("/jpa/users")
  public List<User> retrieveAllUsers()

 {
    return userRepository.findAll();
```

```
 }

   @GetMapping("/jpa/users/{id}")
  public Resource<User> retrieveUser(@PathVariable int id)
{
    Optional<User> user = userRepository.findById(id);

    if (!user.isPresent())
      throw new UserNotFoundException("id-" +

 id);

    // "all-users", SERVER_PATH +

 "/users"
    // retrieveAllUsers
    Resource<User> resource = new Resource<User>
(user.get());

    ControllerLinkBuilder linkTo =
linkTo(methodOn(this.getClass()).retrieveAllUsers());

    resource.add(linkTo.withRel("all-users"));

    // HATEOAS

    return

 resource;
   }

  @DeleteMapping("/jpa/users/{id}")
  public void deleteUser(@PathVariable int id) {
     userRepository.deleteById(id);
```

```
 }

    //
  // input - details of

 user
  // output - CREATED & Return the created URI

  //  HATEOAS

   @PostMapping("/jpa/users")
  public ResponseEntity<Object> createUser(@Valid
@RequestBody User user) {
    User savedUser =  userRepository.save(user);

    URI location =
ServletUriComponentsBuilder.fromCurrentRequest().path("/{id
}").buildAndExpand(savedUser.getId())


 .toUri();

    return  ResponseEntity.created(location).build();

  }

  @GetMapping("/jpa/users/{id}/posts")
  public List<Post> retrieveAllUsers(@PathVariable int id)
{
    Optional<User> userOptional =

 userRepository.findById(id);

    if(!userOptional.isPresent()) {
      throw new UserNotFoundException("id-" +  id);
    }

    return
```

```
 userOptional.get().getPosts();
    }


  @PostMapping("/jpa/users/{id}/posts")
  public ResponseEntity<Object> createPost(@PathVariable

 int id, @RequestBody Post post)  {

    Optional<User> userOptional =
userRepository.findById(id);

    if(!userOptional.isPresent()) {
      throw new UserNotFoundException("id-" + id);
    }

    User user = userOptional.get();



 post.setUser(user);



 postRepository.save(post);

    URI location =
ServletUriComponentsBuilder.fromCurrentRequest().path("/{id
}").buildAndExpand(post.getId())


 .toUri();

    return

 ResponseEntity.created(location).build();

  }
 }
```

/src/main/resources/data.sql Modified

New Lines

```
insert into user values(10001, sysdate(), 'AB');
insert into user values(10002, sysdate(), 'Jill'); insert
into user values(10003, sysdate(), 'Jam');
insert into post values(11001, 'My First Post', 10001);
insert into post values(11002, 'My Second Post', 10001);
```

RESTful Best Practices

Richardson Maturity Model

Level 0

Expose SOAP web services in REST style

- http://server/getPosts
- http://server/deletePosts
- http://server/doThis

Level 1

- Expose Resources with proper URI
  - http://server/accounts
  - http://server/accounts/10

- Improper use of HTTP Methods

Level 2

- Level 1 + HTTP Methods

Level 3

- Level 2 + HATEOAS
  - Data + Next Possible Actions

Best Practices in RESTful Design

- Consumer First
- Make best use of HTTP
  - Request Methods
    - GET
    - POST
    - PUT

- - - DELETE

- Response Status
  - 200 - SUCCESS
  - 404 - RESOURCE NOT FOUND
  - 400 - BAD REQUEST
  - 201 - CREATED
  - 401 - UNAUTHORIZED
  - 500 - SERVER ERROR

- No Secure Info in URI
- Use Plurals
  - Prefer /users to /user
  - Prefer /users/1 to /user/1

- Use Nouns for Resources
- For Exceptions
  - Define a Consistent Approach
    - /search
    - PUT /gists/{id}/star
    - DELETE /gists/{id}/star

- Consumer First
- Define Organizational Standards
  - YARAS - https://github.com/darrin/yaras
    - Naming Resources
    - Request Response Structures
    - Common Features Standardization
      - Error Handling
      - Versioning
      - Searching
      - Filtering
      - Support for Mock Responses
      - HATEOAS

- Build a Framework
- Focus on Decentralized Governance

Bonus Introduction Sections

3 Bonus Sections - Introduction to Spring, Spring Boot and JPA

Title	Category	Github
Spring Framework in 10 Steps	Introduction	Project Folder on Github
Spring Boot in 10 Steps	Introduction	Project Folder on Github
JPA in 10 Steps	Introduction	Project Folder on Github

in28minutes

Become an expert on Spring Boot, APIs, Microservices and Full Stack Development

Checkout the Complete in28Minutes Course Guide