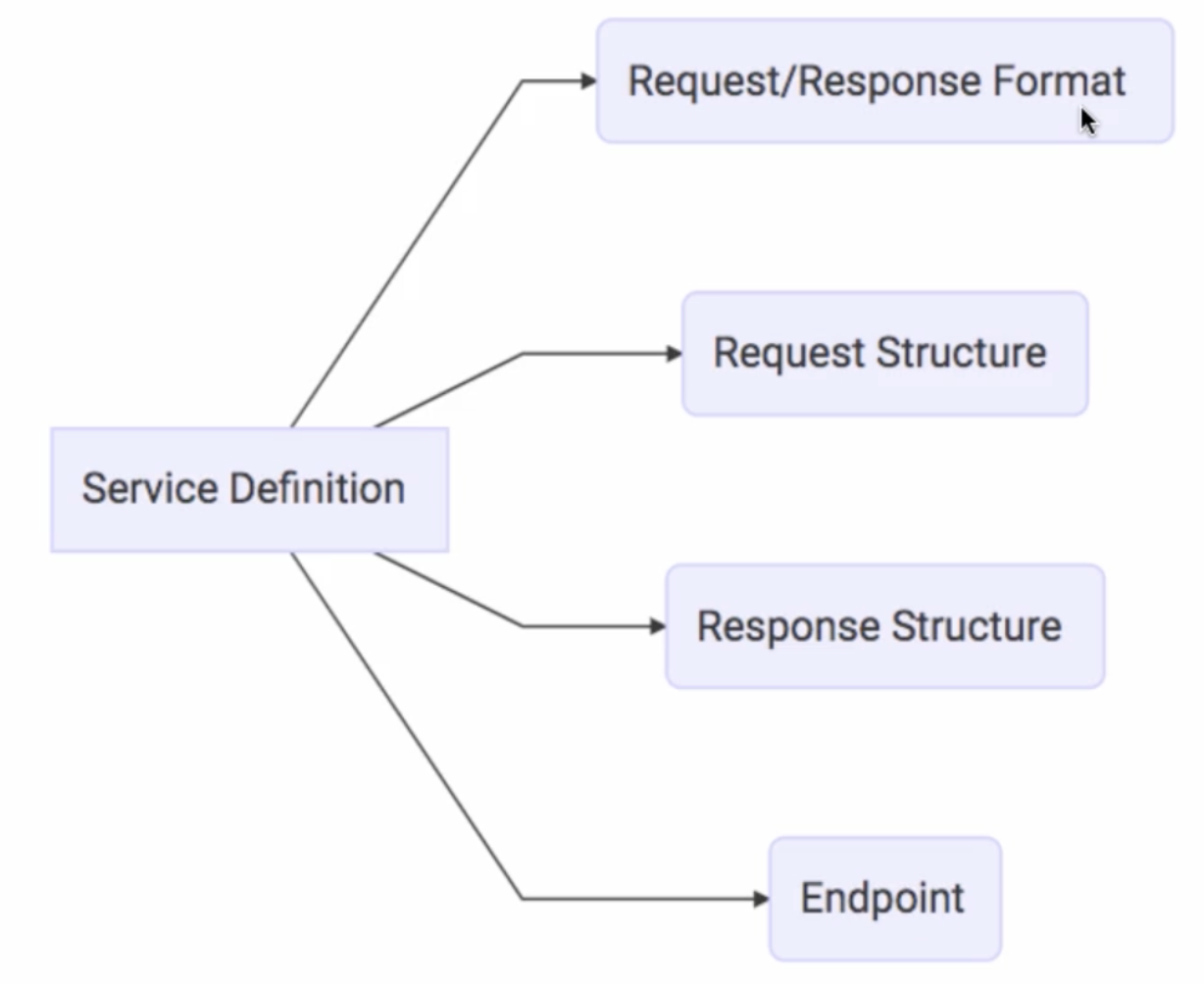
What is webservice?

A service available on the web.

A software system designed to support interoperable machine to machine interaction over a network.

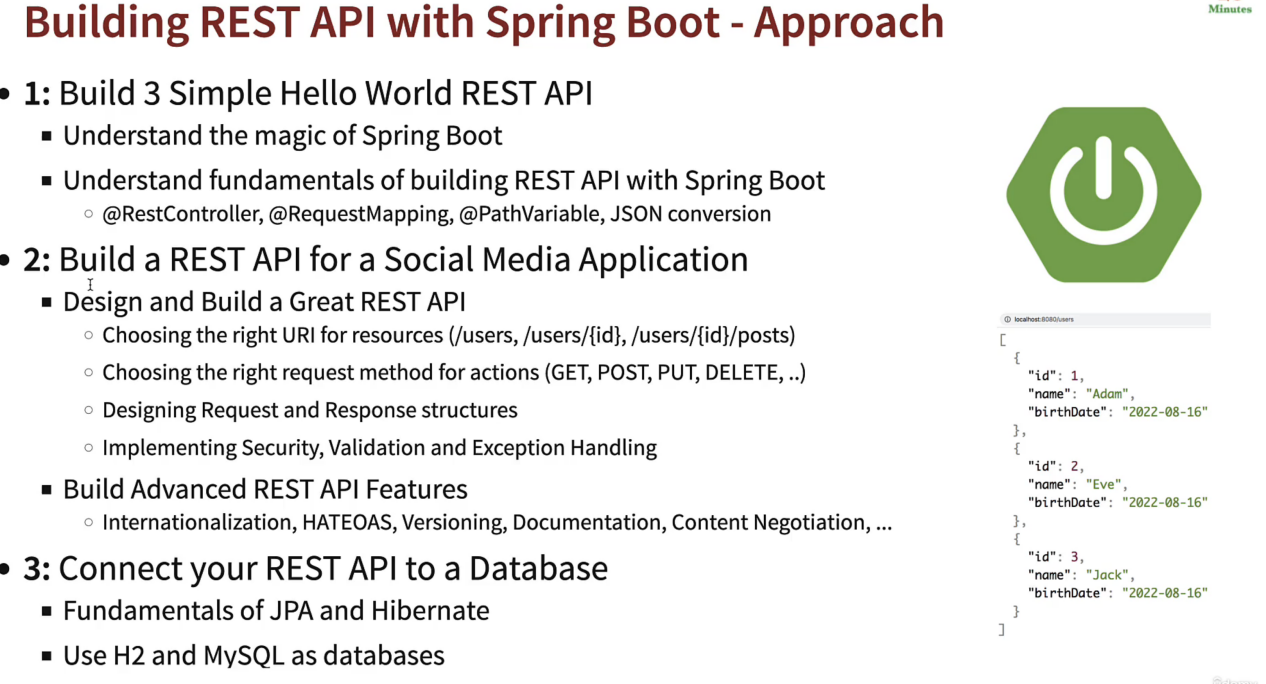


SOAP: Defines specific format for webservices.

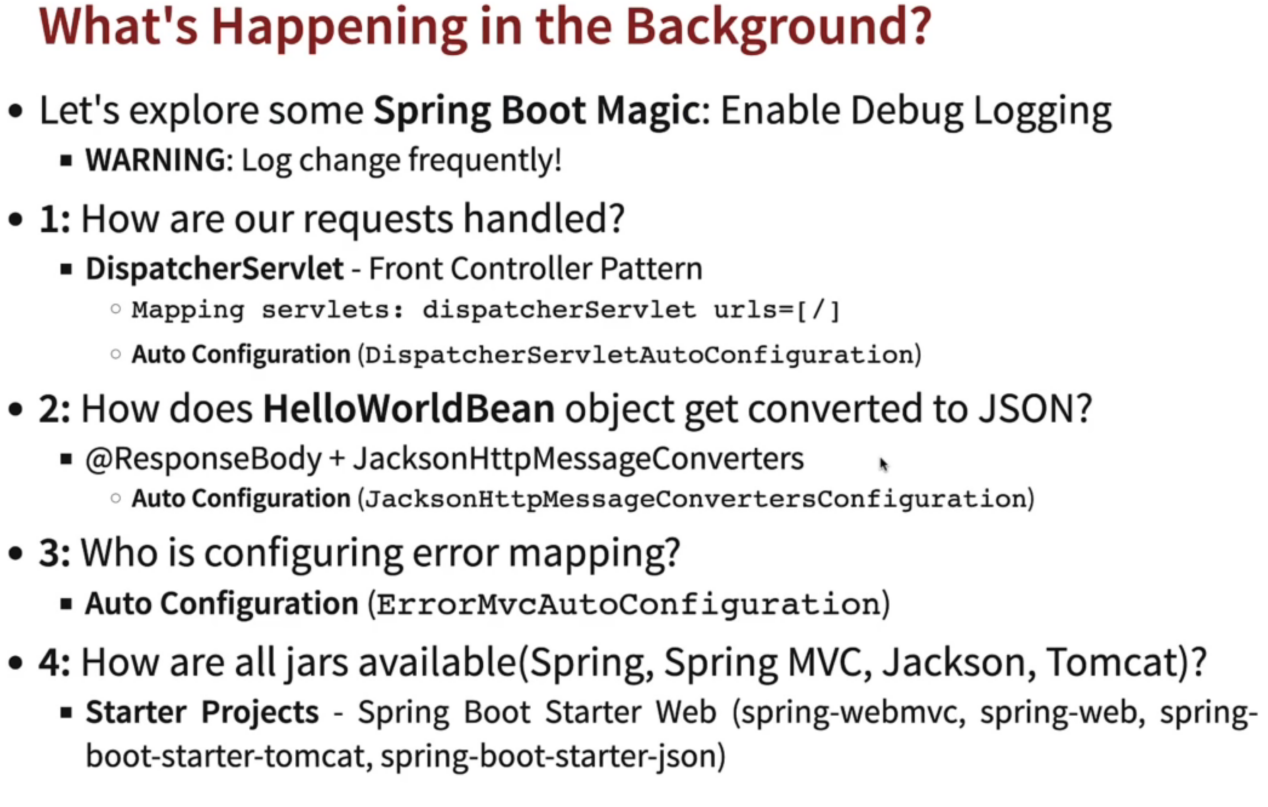
While SOAP and REST share similarities over the HTTP protocol, **SOAP is a more rigid set of messaging patterns than REST**. The rules in SOAP are important because we can't achieve any level of standardization without them. REST as an architecture style does not require processing and is naturally more flexible

Soap Vs Restful webservices:

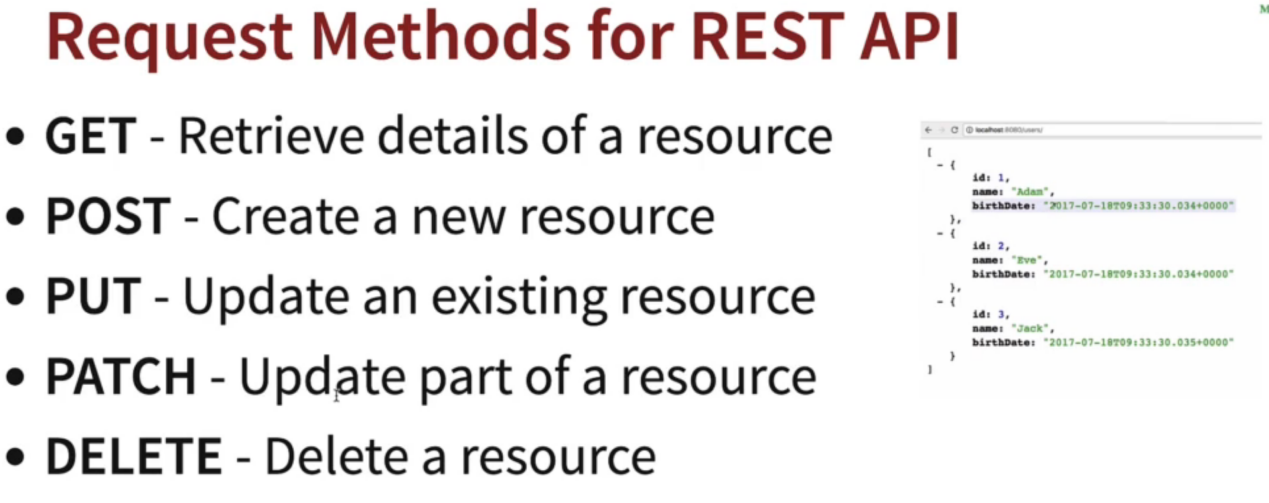
|  |  |  |
| --- | --- | --- |
| ****SOAP**** | ****REST**** | |
| SOAP stands for Simple Object Access Protocol | REST stands for Representational State Transfer | |
| SOAP is a protocol. SOAP was designed with a specification. It includes a WSDL file which has the required information on what the web service does in addition to the location of the web service. | REST is an Architectural style in which a web service can only be treated as a RESTful service if it follows the constraints of being   1. Client Server 2. Stateless 3. Cacheable 4. Layered System 5. Uniform Interface | |
| SOAP cannot make use of REST since SOAP is a protocol and REST is an architectural pattern. | REST can make use of SOAP as the underlying protocol for web services, because in the end it is just an architectural pattern. | |
| SOAP uses service interfaces to expose its functionality to client applications. In SOAP, the WSDL file provides the client with the necessary information which can be used to understand what services the web service can offer. | REST use Uniform Service locators to access to the components on the hardware device. For example, if there is an object which represents the data of an employee hosted on a URL as http://demo.guru99 , the below are some of URI that can exist to access them.  http://demo.guru99.com/Employee  http://demo.guru99.com/Employee/1 | |
| SOAP can only work with XML format. As seen from SOAP messages, all data passed is in XML format. | REST permits different data format such as Plain text, HTML, XML, JSON, etc. But the most preferred format for transferring data is JSON. | |
| SOAP requires more bandwidth for its usage. Since SOAP Messages contain a lot of information inside of it, the amount of data transfer using SOAP is generally a lot.  <?xml version="1.0"?>  <SOAP-ENV:Envelope  xmlns:SOAP-ENV  ="http://www.w3.org/2001/12/soap-envelope"  SOAP-ENV:encodingStyle  =" http://www.w3.org/2001/12/soap-encoding">  <soap:Body>  <Demo.guru99WebService  xmlns="http://tempuri.org/">  <EmployeeID>int</EmployeeID>  </Demo.guru99WebService>  </soap:Body>  </SOAP-ENV:Envelope> | | REST does not need much bandwidth when requests are sent to the server. REST messages mostly just consist of JSON messages. Below is an example of a JSON message passed to a web server. You can see that the size of the message is comparatively smaller to SOAP.  {"city":"Mumbai","state":"Maharastra"} |



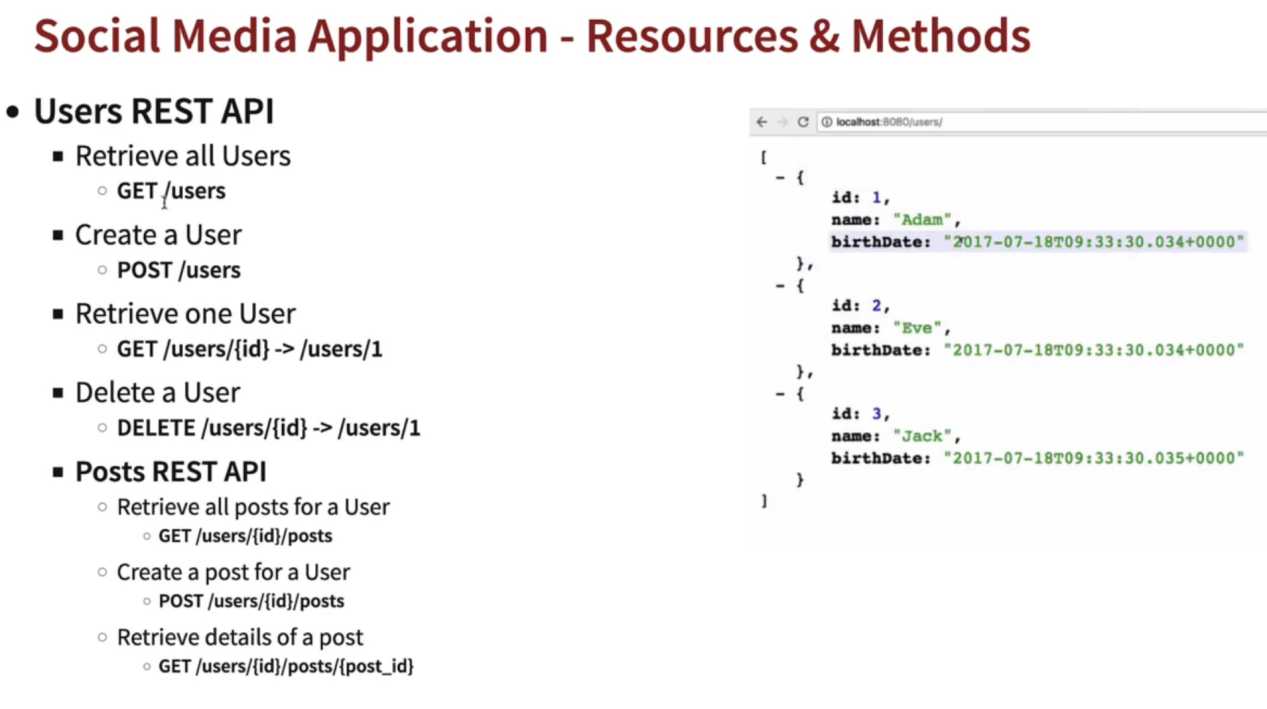
Example1: HelloWorld Program

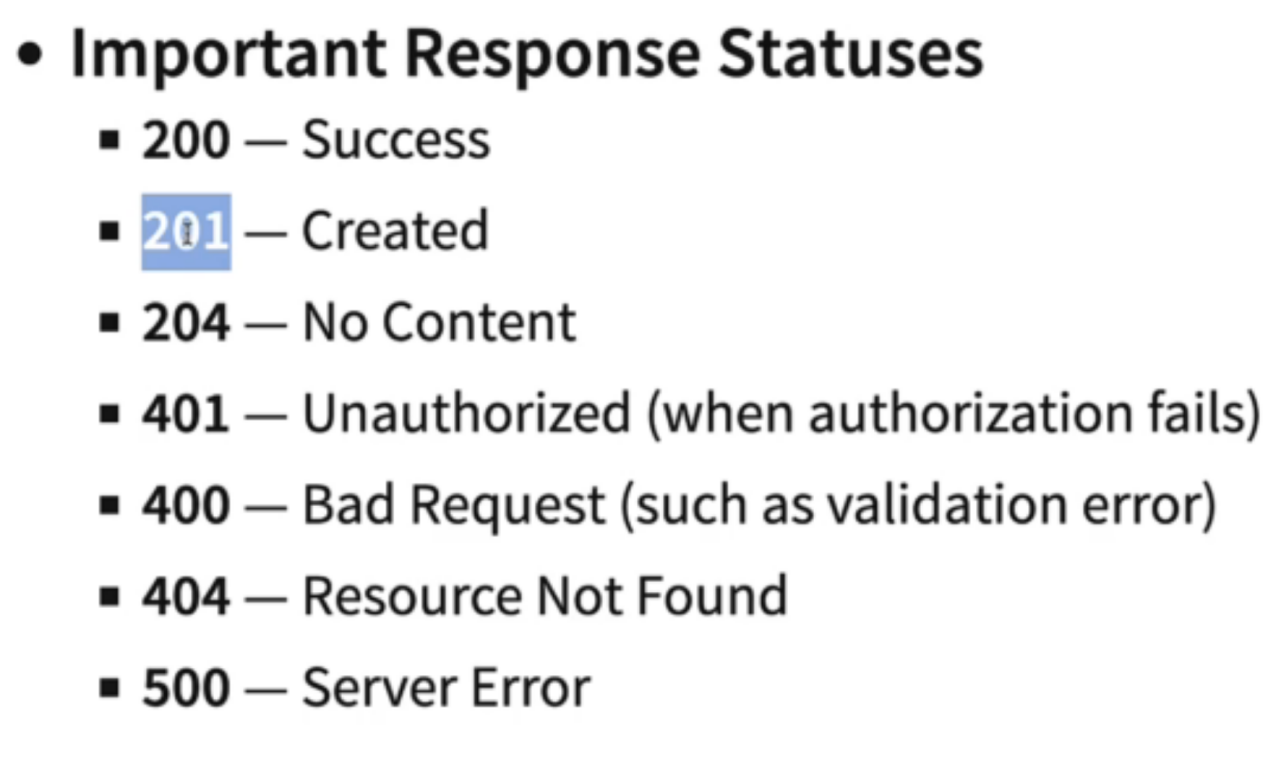


Example 2: Helloworld using pathvariable



Build social medial application:





if we want to return response code with post method we need to use ResponseEntiry.created(url).build()

Example : GetMapping & PostMapping functions with static data

Example: GenericExceptionHandler in Springboot

**Global Exception Handling in Spring boot:**

1. If you want to define exception details with properties like exception message, timestamp & error details then define one Exception Bean with these properties. Generate consturctor with parameters.
2. create a class which extends ResponseEntityExceptionHandler

This super class has all the pre defined methods for exception handling. If you want to use , you can reuse or you can over ride those methods. If you want to define new function you can copy signatures of predefined methods.

1. Mark this class as @ControllerAdvice. so all the exceptions in spring boot application be redirected to this class.
2. Define a function which handles that exception

-> method should be declared with @ExceptionHandle(Exception.class)

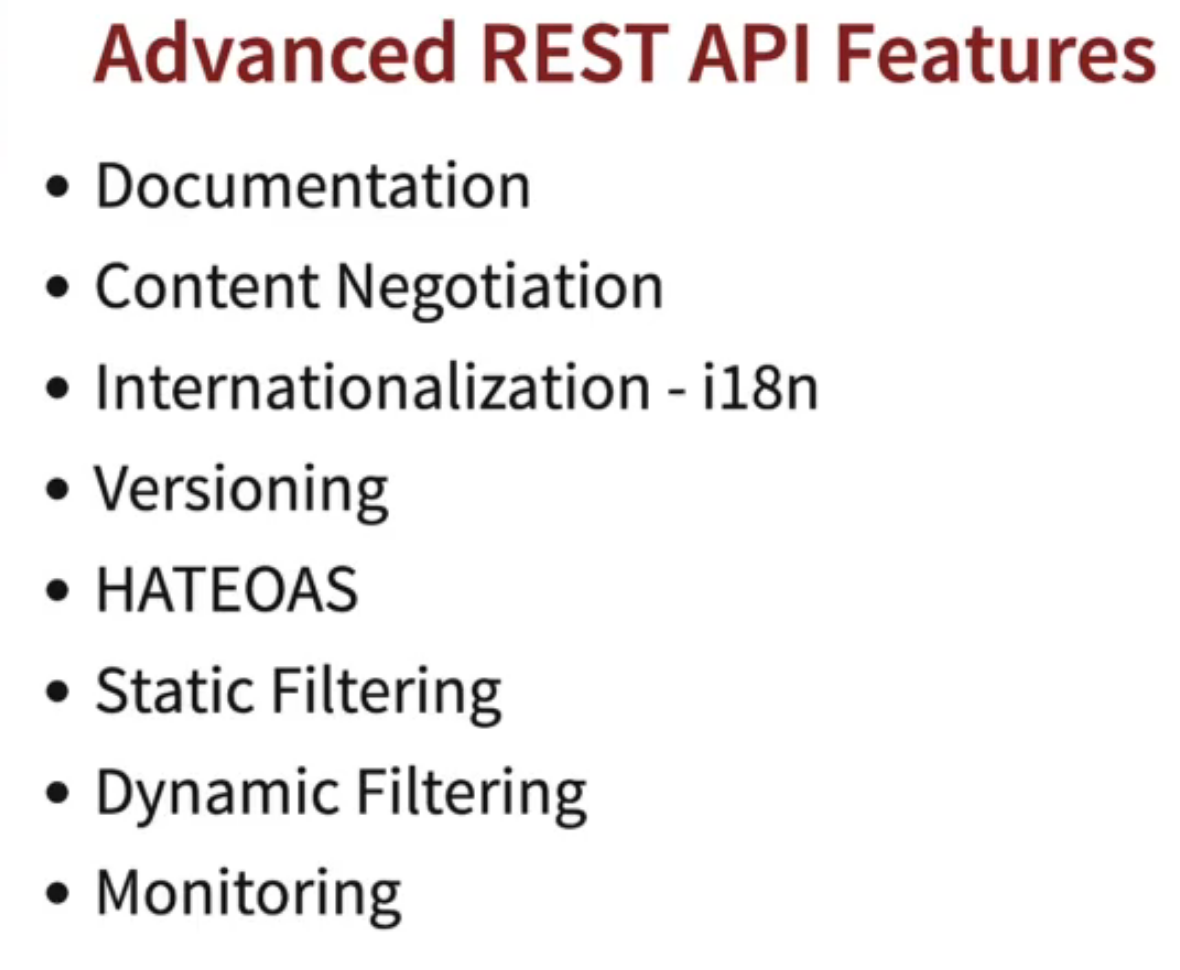
you need to pass specific exception class which you want to handle in Exception Handler annotation.

1. with in the method you need pass arguments of Exception Bean constructor & return response in the format of Response Entity.add exceptionbean object in response entity.
2. If you want to return response code as well then include response code as HTTPSTATUS.NOT\_FOUND in response.
3. In restcontroller where ever you are throwing customized exception those are redirected to controller advice class for exception handlers.

Example in exception package of restful webservice project.

Example: deleteMapping

Example: validation of properties



**Swagger UI:**

To enable swagger UI / open api documentation add open api dependency in pom.xml

1. <dependency>
2. <groupId>org.springdoc</groupId>
3. <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
4. <version>2.0.0</version>
5. </dependency>

**Content Negotiation:**

By default Rest API response would be in JSON. If user wants response either in xml format or in other formats then we can change response formats by enabling jackson data formats dependency in pom.xml file

<dependency>

<groupId>com.fasterxml.jackson.dataformat</groupId>

<artifactId>jackson-dataformat-xml</artifactId>

<version>2.14.1</version>

</dependency>

Add jackson dataformat dependency in pom.xml, restart server. In postman add header accept : application/xml then you wll get xml response

**I18N:**

If we want to return response in different languages then we can use I18N.

1. define an API for Internationalization in RestController
2. define localization messages in messages.properties

we need to define message with key values in message.properties file

hello.message=Hi! Welcome to Restful Webservices

1. we need to read this message in rest controller

define MessageSource in Restcontroller

*@Autowired*

private MessageSource messageSource;

1. call message key using MessageSource Object.

*@GetMapping*("/hello-world-internationalization")

public String printHelloWorldInternationalization() {

Locale locale = LocaleContextHolder.*getLocale*();

return messageSource.getMessage("hello.message", null, "Defaut message", locale);

}

1. This is for default message. Lets define messages for specific languages.
2. create messages\_nl.properties file for netharlands
3. define message key in that. add translated message for dutch language.
4. In postman add header accept-language : nl & hit api. It will give response in dutch language.

Example in helloworld package of restful webservice project.

**Versioning:**

whenever you are implementing new changes for existing API’s , create new API as version2. so that in future all rest api consumers will shift from version 1 to version 2.

Example in versioning package

basic versioning

versioning with request param

versioning with request header

versioning with content negotiation

**HATEOAS:**

HATEOAS stands for **Hypermedia as the Engine of Application State**and it is a component of RESTful API architecture and design. With the use of HATEOAS, the client-side needs minimal knowledge about how to interact with a server. This is made possible by the network application responding to the client’s requests with dynamically generated information through the use of hypermedia.

Add hateoas dependency in pom.xml file.

If you want to include any links in your response, we can include through hateoas.

If you want to change any attribute name in JSON response you can change using @JsonProperty

*@Size*(min=2,message="Name should be atleast 2 letters")

*@JsonProperty*("user\_name")

private String name;

**Static Filtering:**

Example in filterin pakage

If we want to skip/ignore any property in the response we can use @JsonIgnore for that property.

private String field1;

*@JsonIgnore*

private String field2;

private String field3;

field2 would be skipped in the json response.

we can define json ignore in class level as well. we need to use

@JsonIgnoreProperties({“field1”,”field2”})

*@JsonIgnoreProperties*({"field1","field2"})

public class SomeBean {

private String field1;

private String field2;

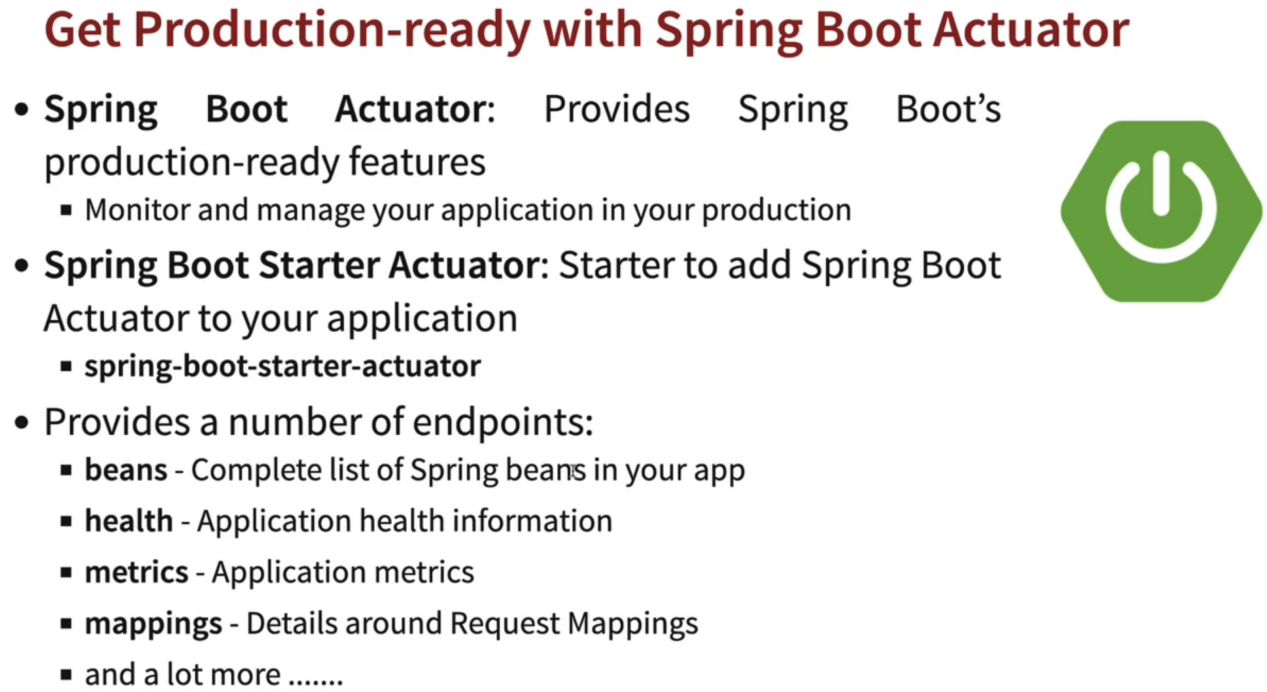
private String field3;

**Dynamic Filtering:**

To dynamicall add filter for the class.

Example in Filtering package

**SpringBoot Acutator:**



**To Enable Springboot Actuator:**

1. Add Actuator dependency in pom.xml file
2. add below in application.properties file.

logging.level.org.springframework=info

management.endpoint.web.exposure.include=\*

1. hit url <http://localhost:8080/actuator>

**HAL Explorer:**



Add Hal Explorer dependency in pom.xml

<dependency>

<groupId>org.springframework.data</groupId>

<artifactId>spring-data-rest-hal-explorer</artifactId>

</dependency>

Hit url : <http://localhost:8080/explorer/index.html#>

#### Enabling H2-Console

Add this in application.properties file

spring.h2.console.enabled=true

**Working with JPA & H2 database:**

**Example in jpa folder**

1. create a bean class and mark it as @Entiry. Its represents DB Table.

*@Entity*(name="user\_details")

public class User {

*@Id*

*@GeneratedValue*

private Integer id;

*@Size*(min=2,message="Name should be atleast 2 letters")

*@JsonProperty*("user\_name")

private String name;

*@Past*(message="DOB should not be in future")

private LocalDate dob;

1. create an interface which extends JPA Repository

public interface UserRepository extends JpaRepository<User,Integer>{

}

1. In Restcontroller autowire repository object. use object to communicate with DB functions

* @Autowired
* private UserRepository repository;
* @GetMapping("/jpa/users")
* public List<User> getUsers(){
* return repository.findAll();
* }

**Spring Security:**

Add spring boot starter security in your pom.xml file

add username & pwd in application.properties file

spring.security.user.name=username

spring.security.user.password=password

In Postman under authorization pass username & pwd. hit API

Example in spring security package