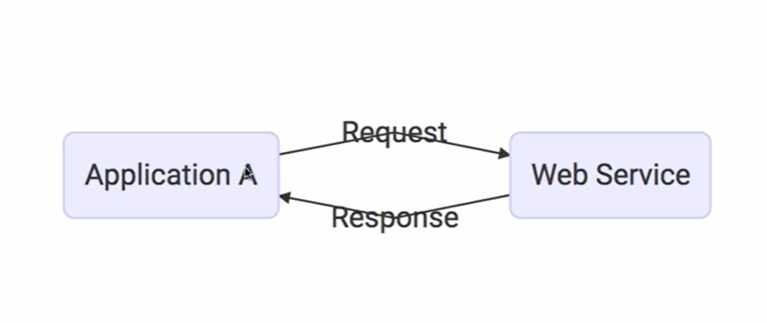
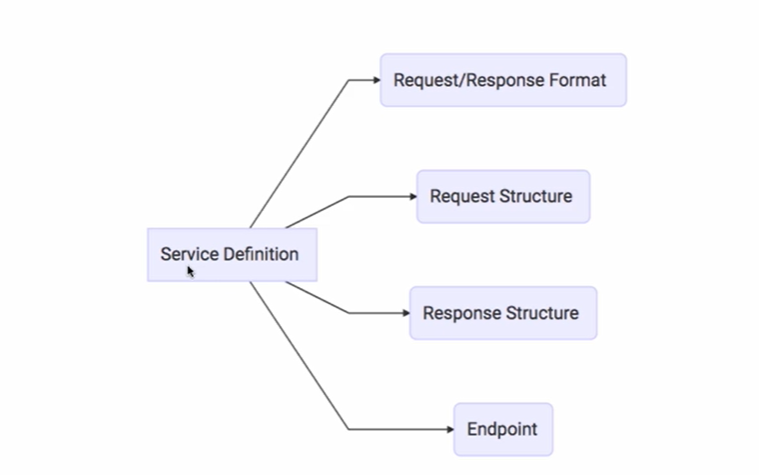
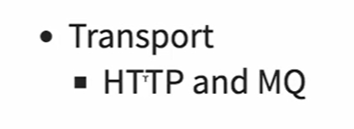
**WEB SERVICES**

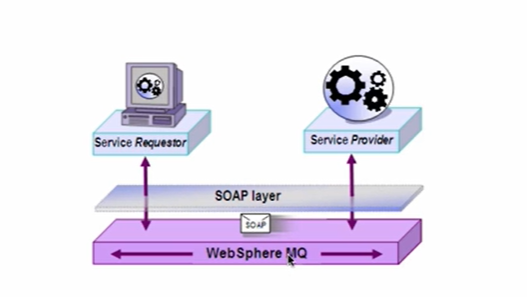
Software system designed to support interoperable machine-to-machine interaction over a network.



**Request and Response format : XML (Extensible Markup Language) , JSON (Java Script Object Notation)**Every webservice has service definition.



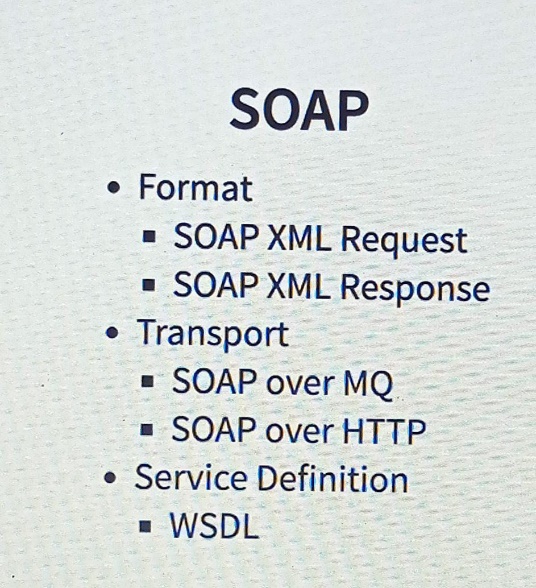


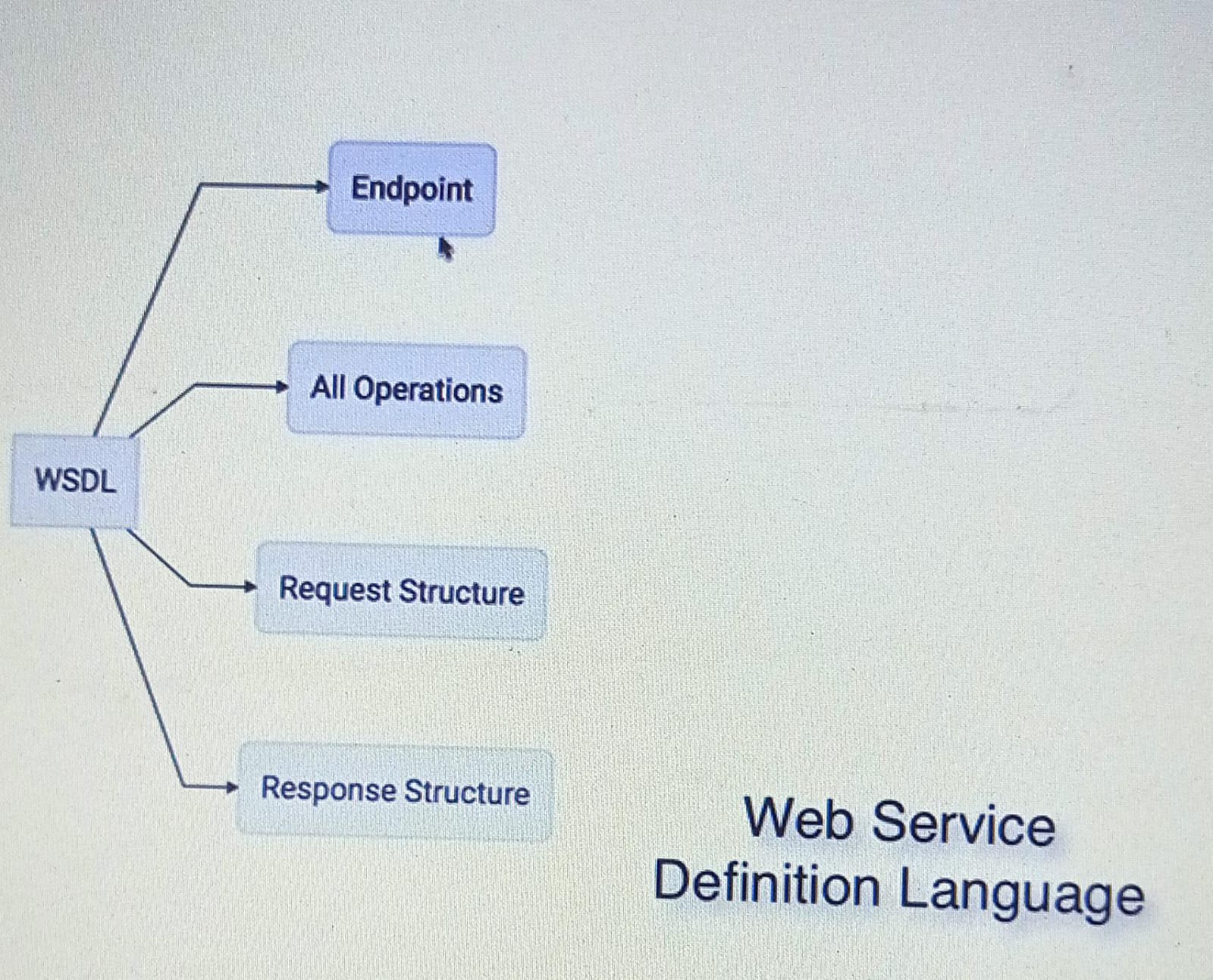


**TYPES OF WEBSERVICES**

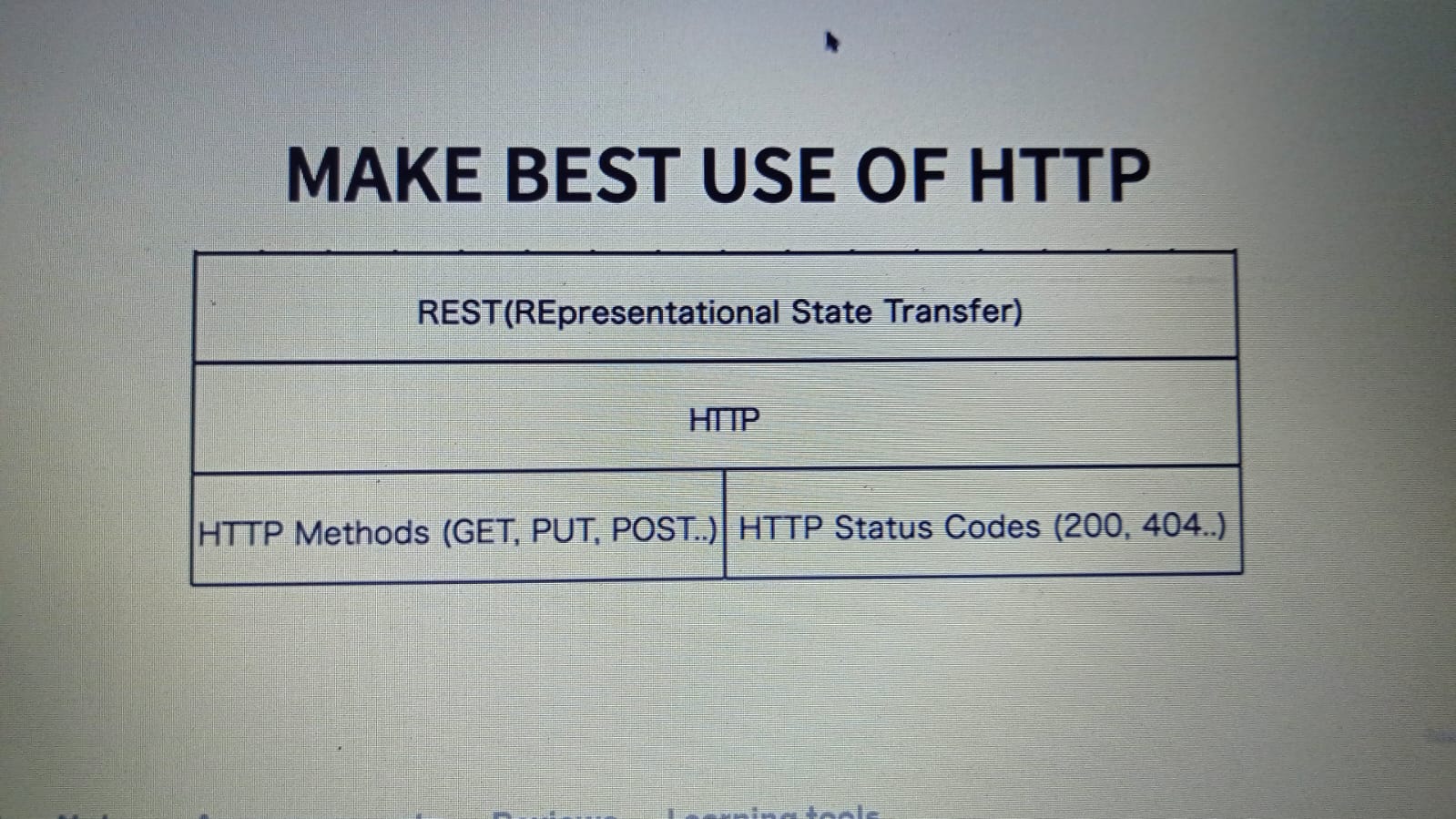
1**.**SOAP  
2.RESTFUL

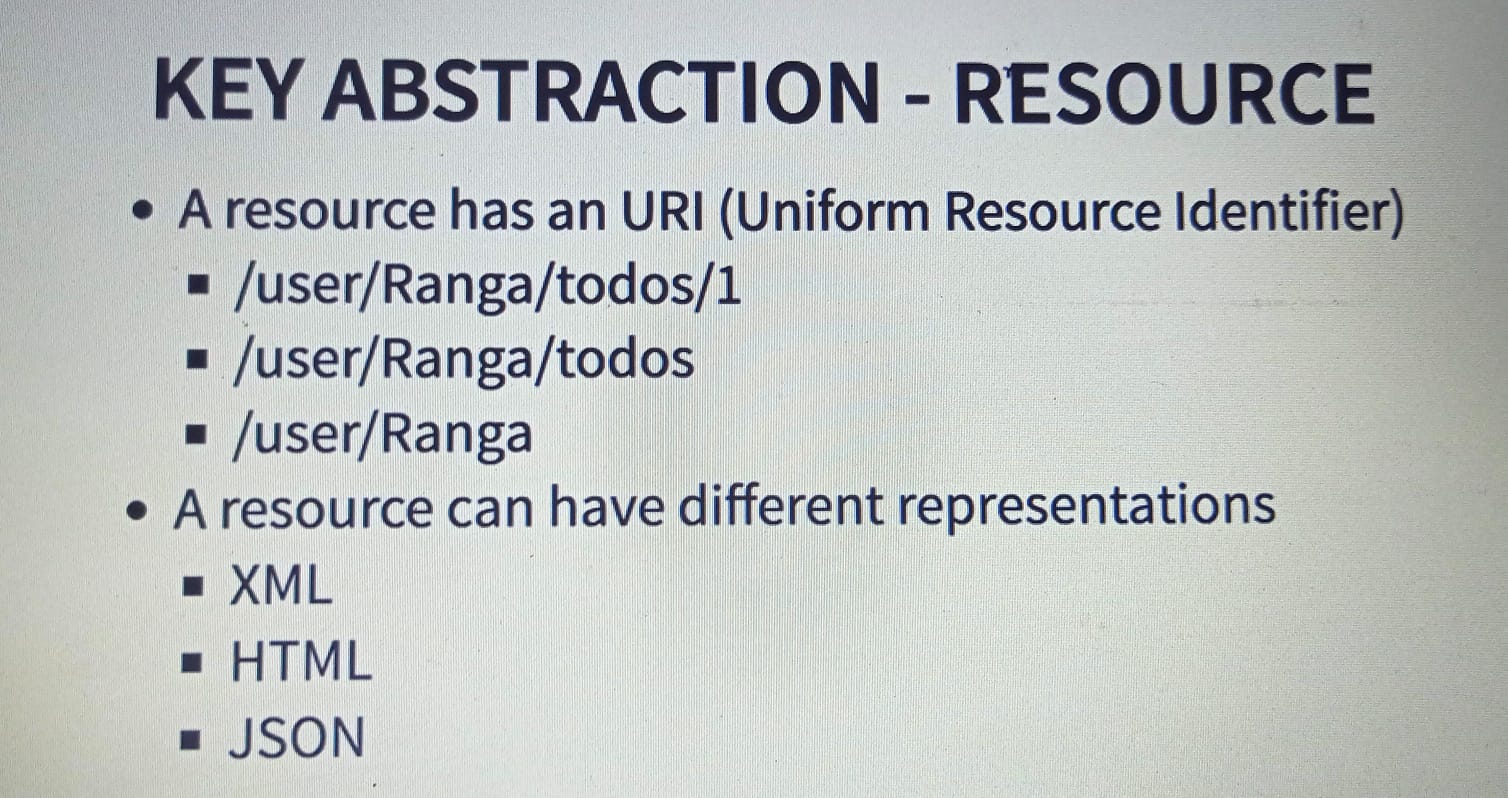
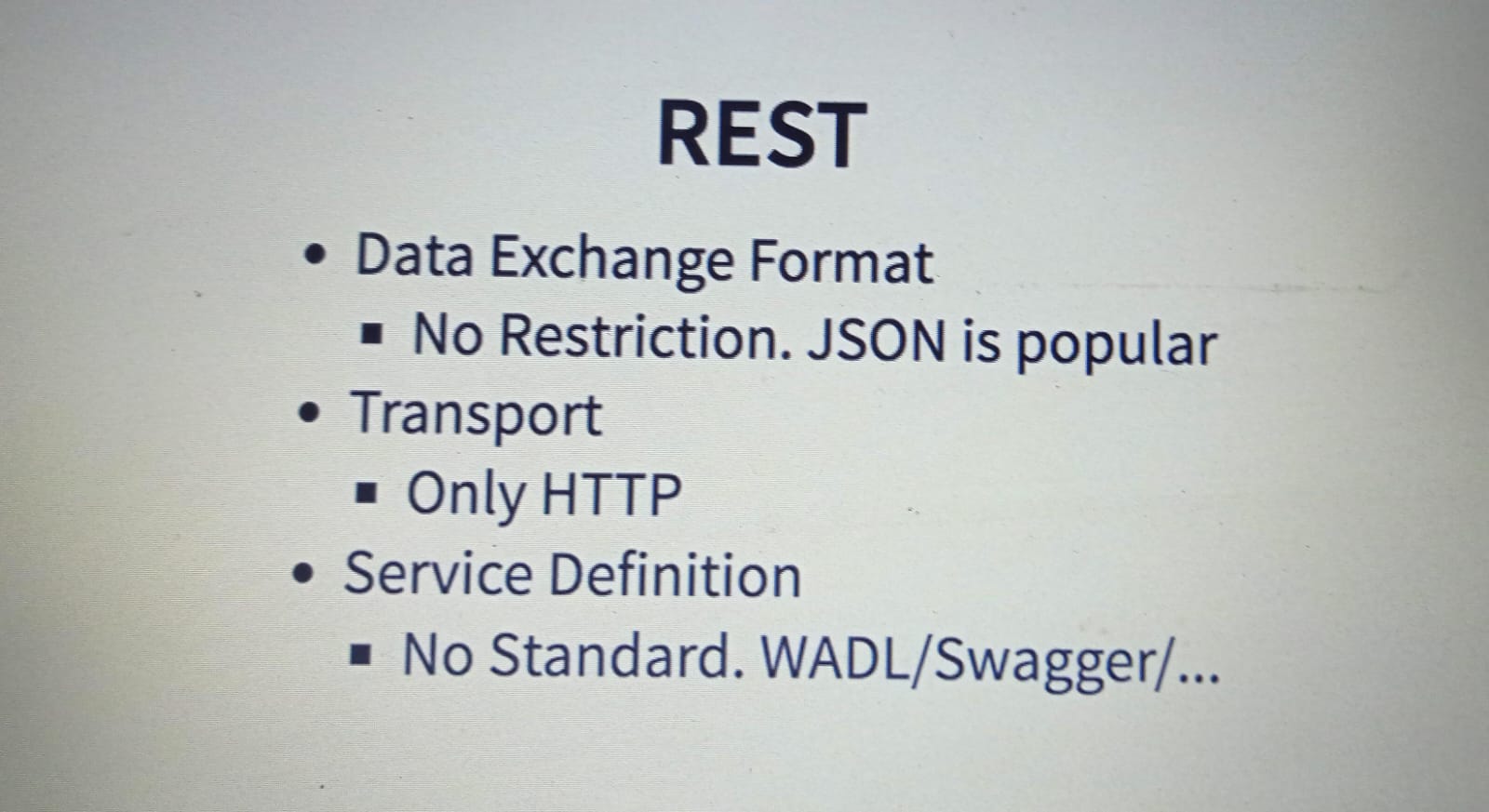
**1.SOAP – Simple Object Access Protocol  
XML format  
eg of XML format :  
<getCourseDetailsRequest>  
 <id>Course1</id>  
</getCourseDetailsRequest>**

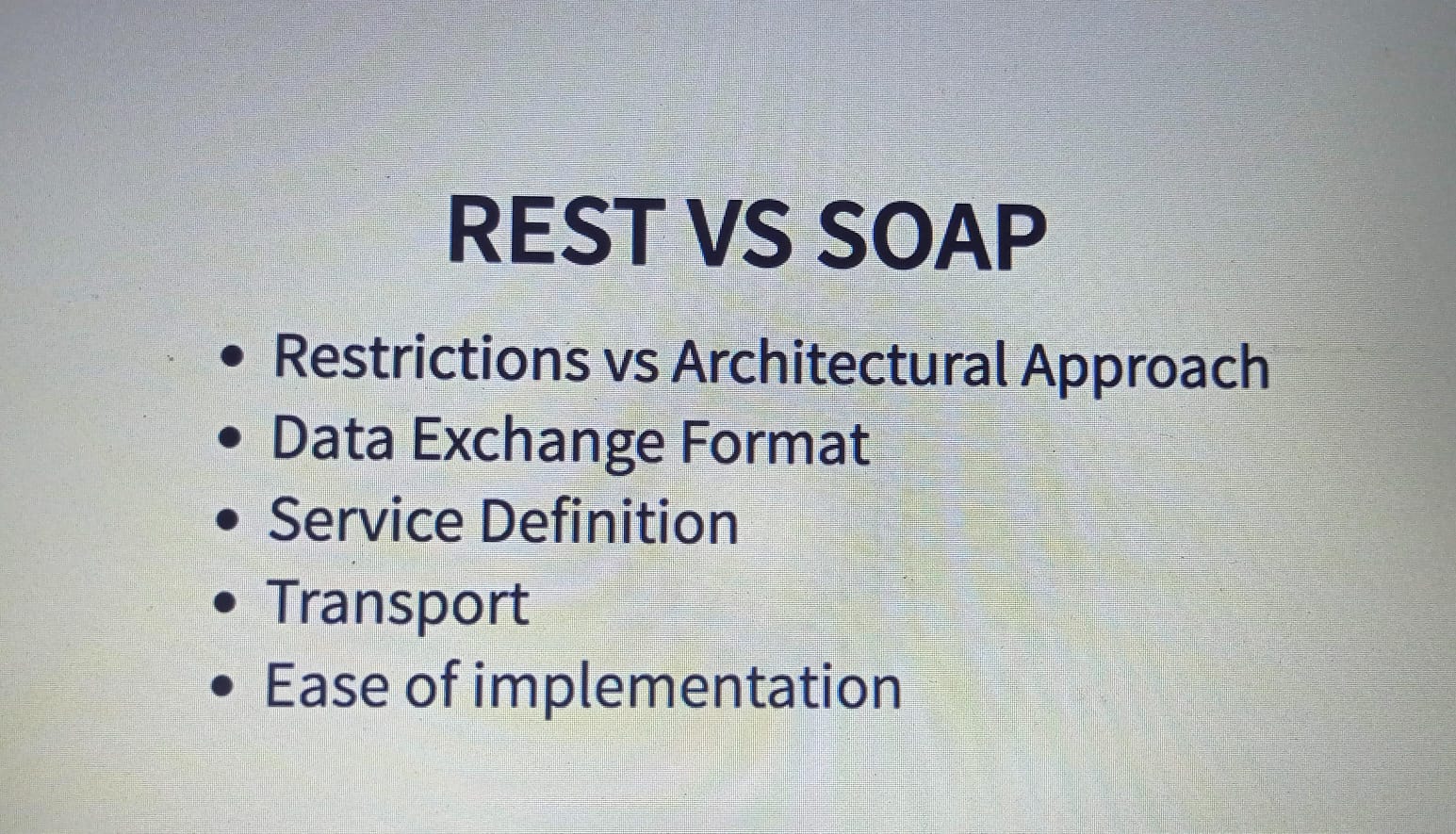
****

****

**2.REST – Representational State Transfer**

**HTTP – Hyper Text Transfer Protocol  
**

****

****

**April 29, 2024**

**Sample controller :** @RequestMapping(method = RequestMethod.***GET***, path = "/hello-world" )

==

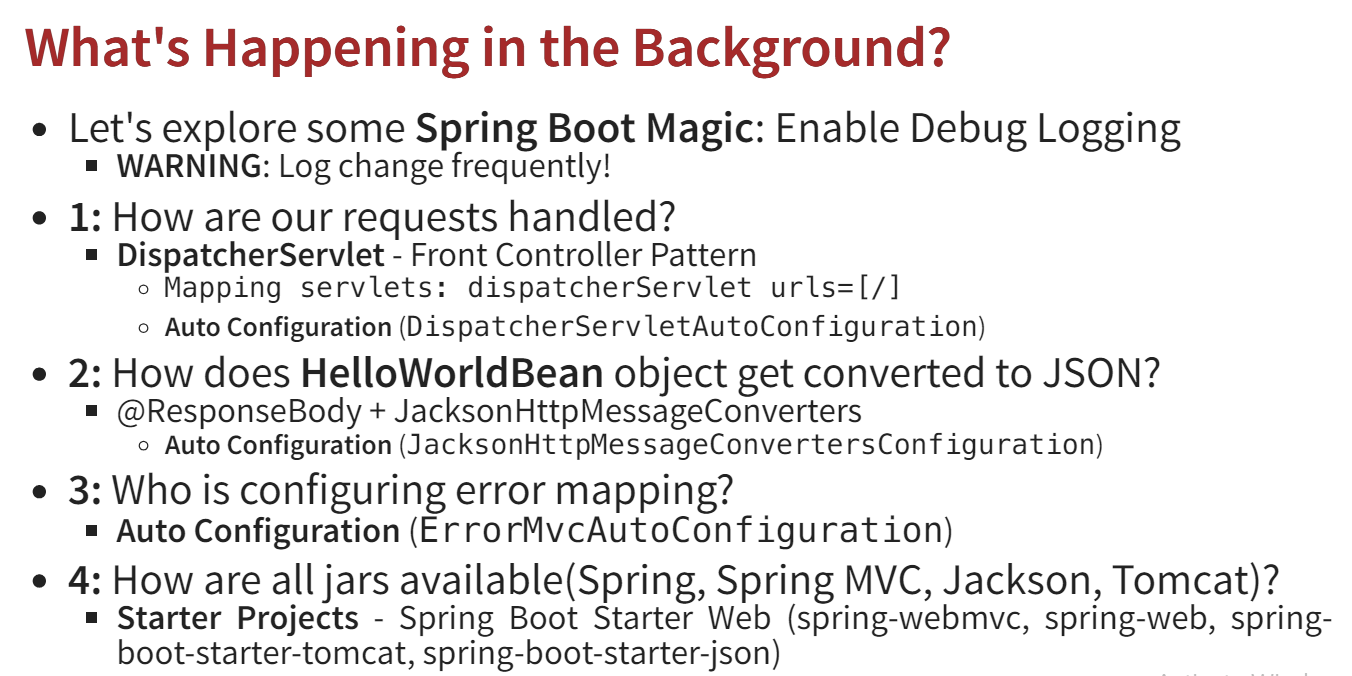
@GetMapping(path = "/hello-world")

==

@GetMapping("/hello-world")

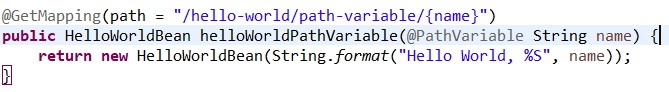
To print logs in console at debug level use below in application.properties file

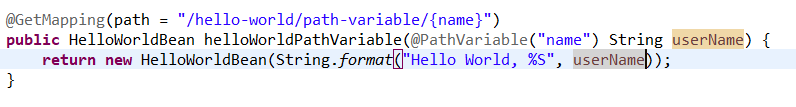
**logging.level.org.springframework = debug**

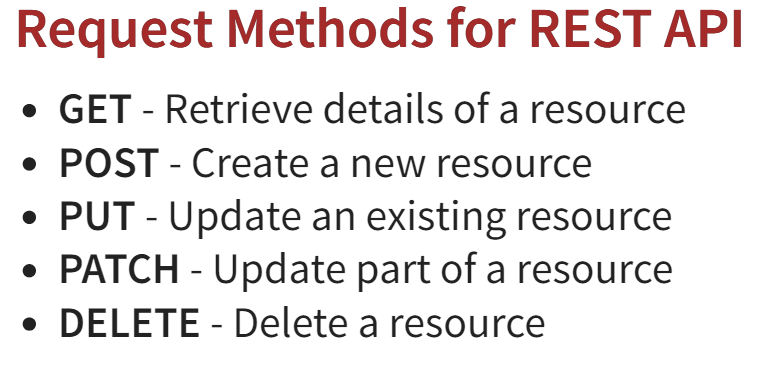


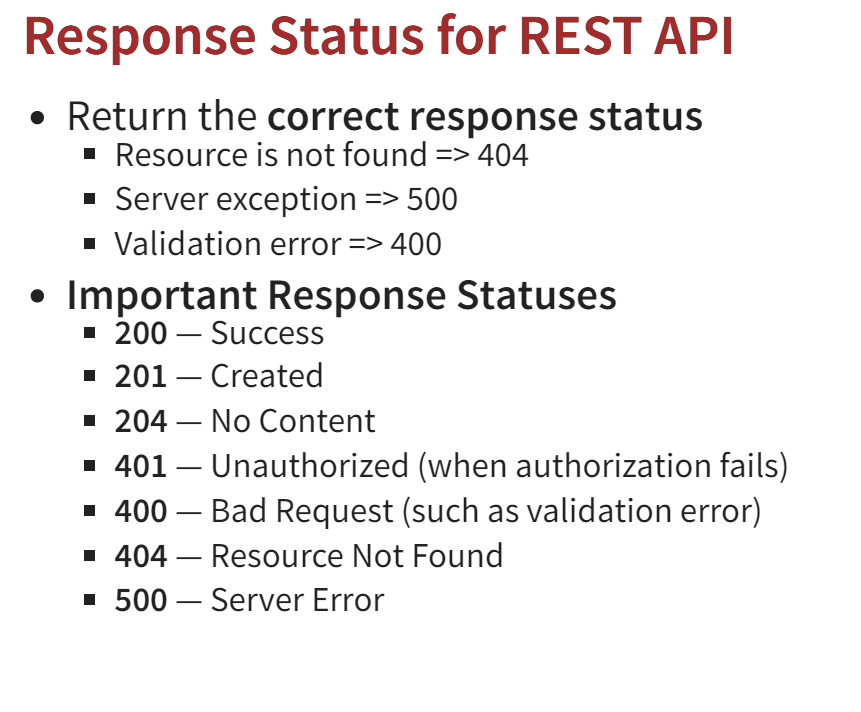
**What is autoconfiguration ?**

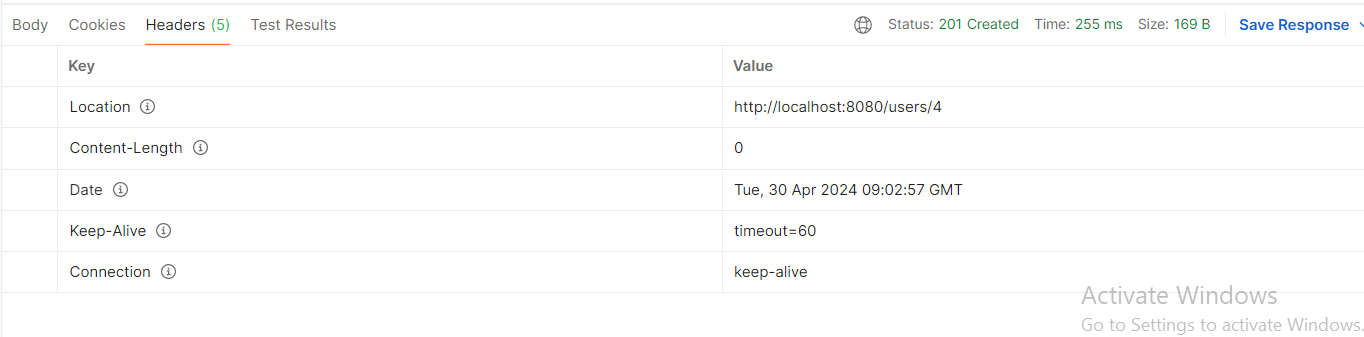
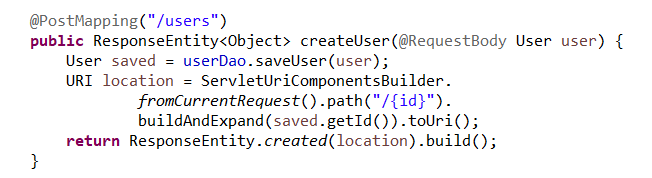
**Based on jars available in the classpath it will autoconfigure all the things.**

**Path parameters or PathVariable:**



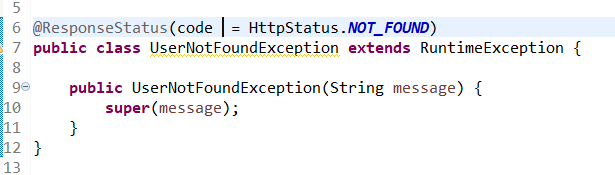






**April 30, 2024**

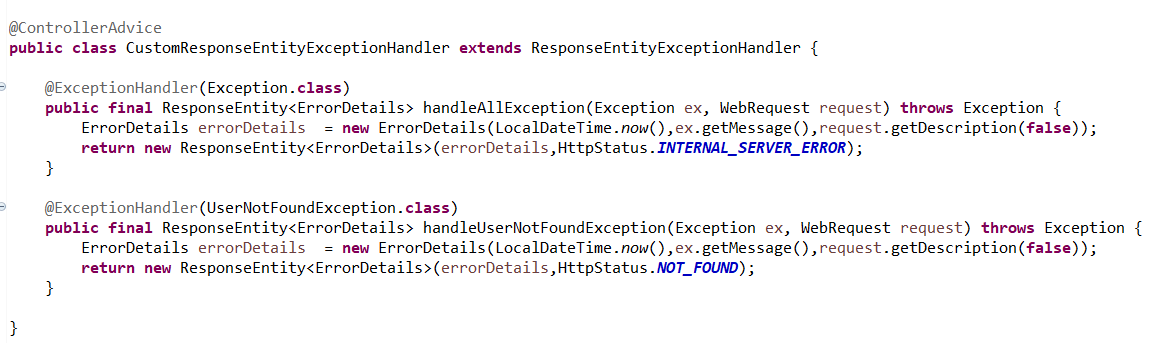
Custom exception with custom status code



**in prod you will run the jar file as java -jar filename then dev tools will be automatically disabled.**

@ControllerAdvice is used to create a global exception handler that returns a view (HTML) as the response. It is typically used in web applications where you want to display a custom error page to the user when an exception occurs.

Eg:



To perform validations, you need to have **spring-boot-starter-validation** dependency.

Seen demo on validations and custom validation messages displaying.

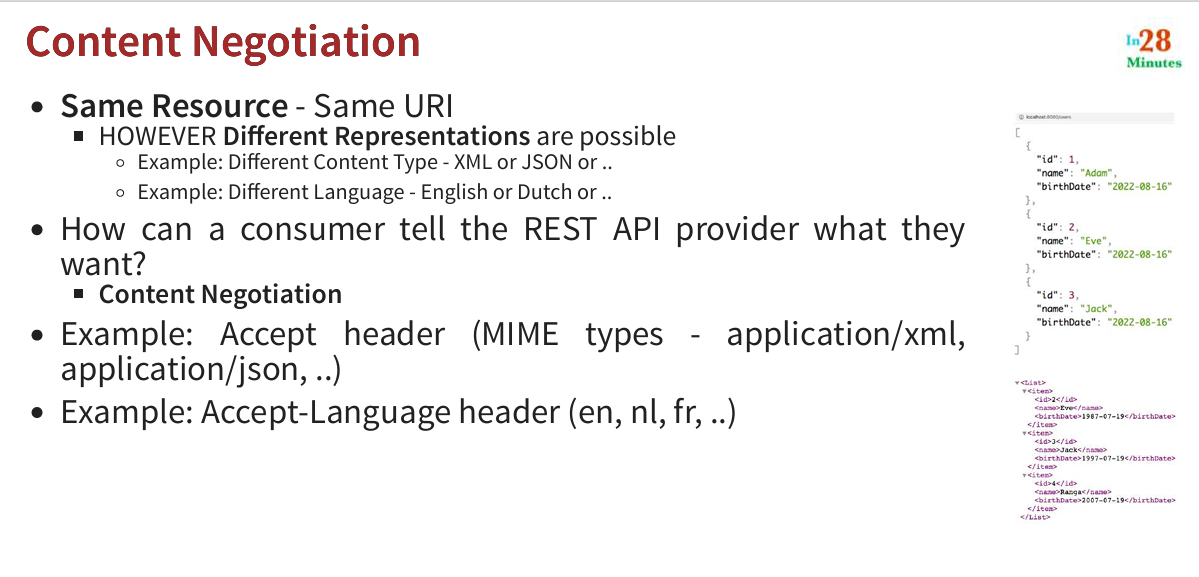


Seen demo on how to use **springdoc-openapi-starter-webmvc-ui**

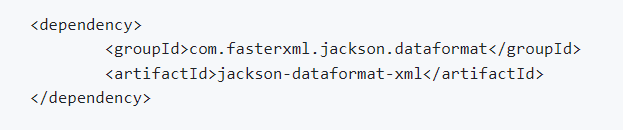
dependency for generating documentation for our rest api.(**swagger** ui also seen)[**http://localhost:8080/swagger-ui**](http://localhost:8080/swagger-ui)**.html**[**http://localhost:8080/v3/api-docs**](http://localhost:8080/v3/api-docs)

**May 1, 2024**

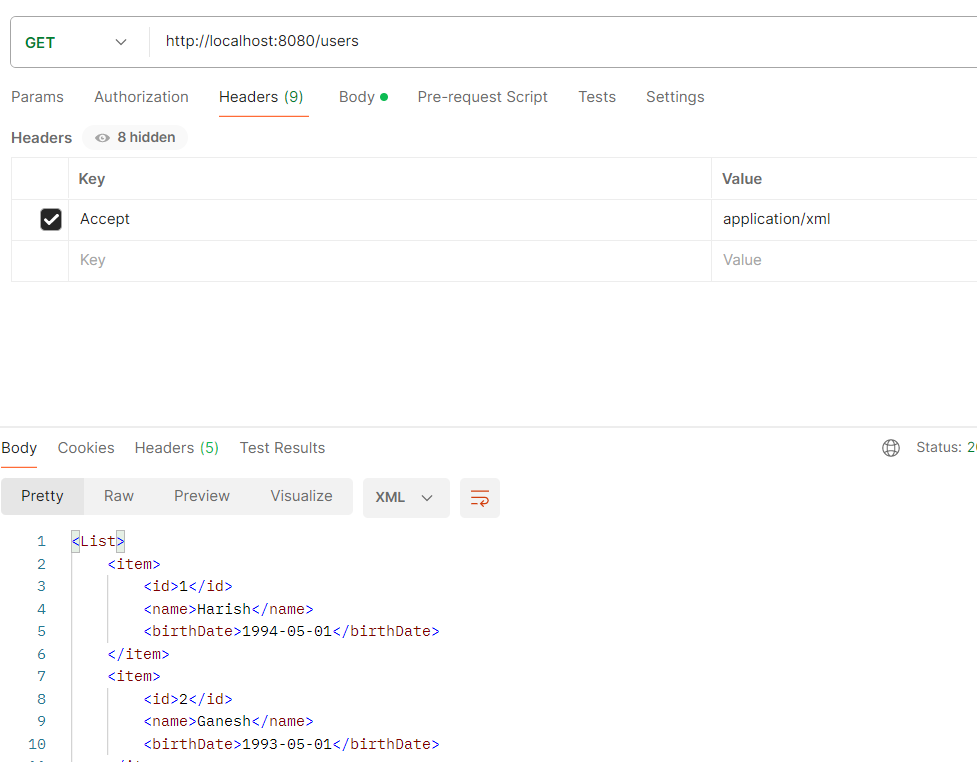
**CONTENT NEGOTIATION**



If you want your response in xml format then follow the below steps:



Add this dependency in pom.xml

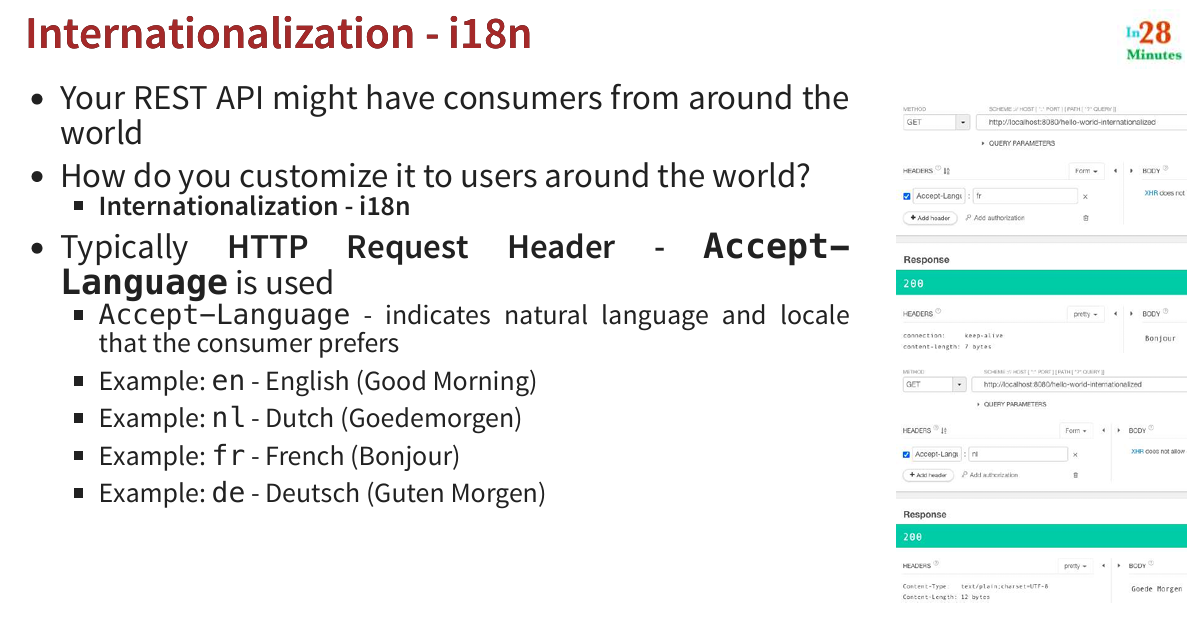


Add Headers as above in postman and you will get the response in xml format.

Standard way to define **internationalization** (i18n) is to have it in messages.properties.

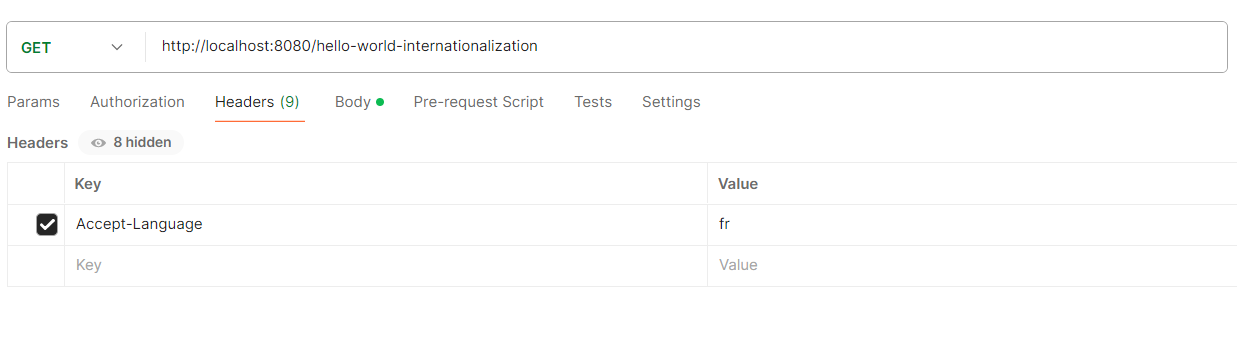
For other language like dutch you need to create **messages\_nl.properties**.

Lly for other as well.

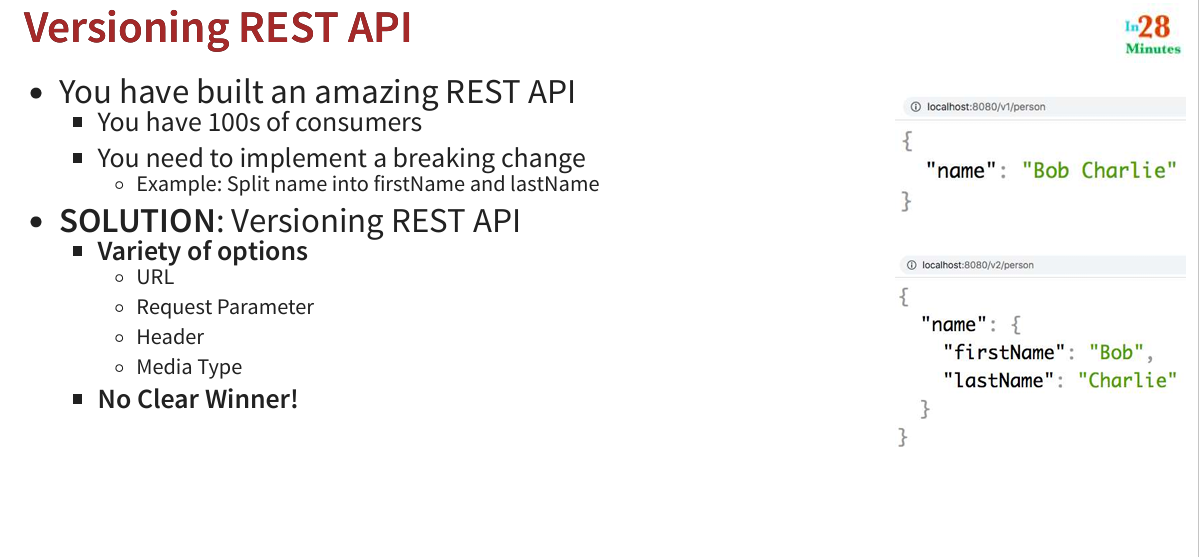


For code reference check the github url.

To get response from French you need to give key as Accept-Language and value as fr.

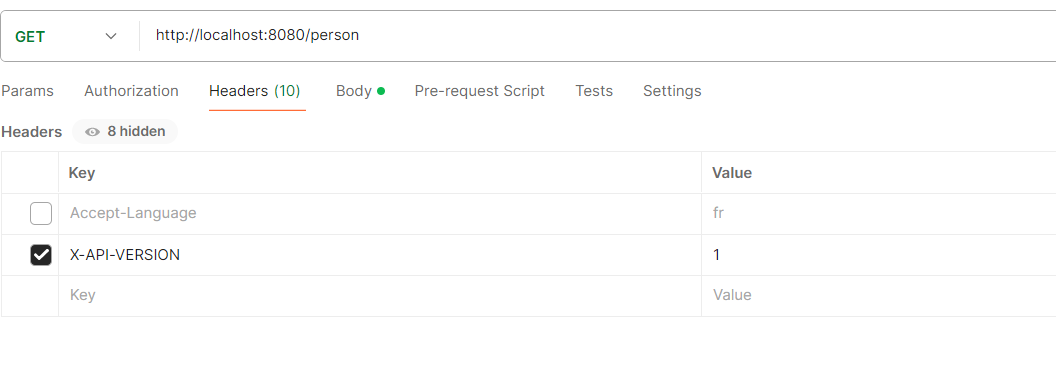


**Versioning**

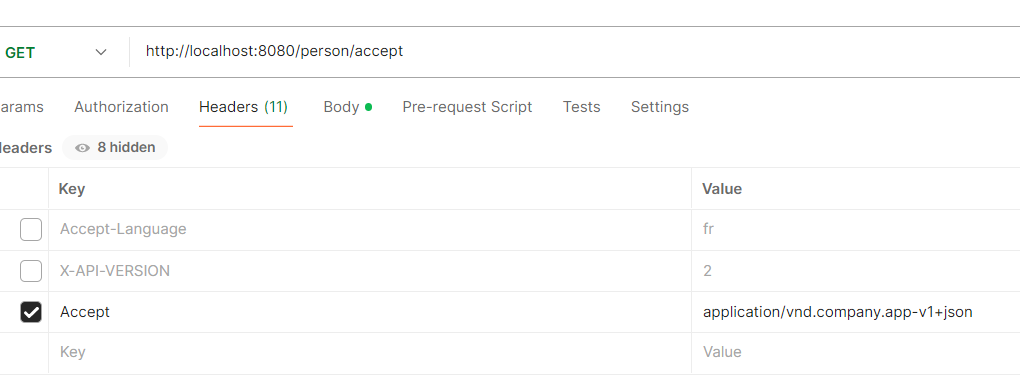


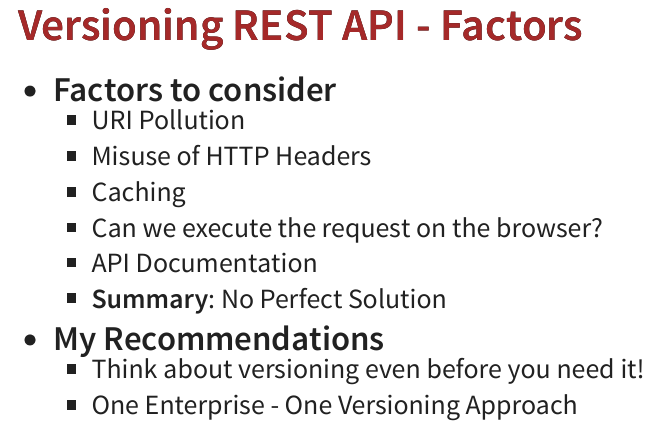


**For Custom headers versioning use below :**



**For media type versioning use below:**





**Factors to choose for versioning:**

**1.URI Pollution**

Uri versioning and request parameter versioning will pollute the url.

**2.Missue of HTTP Headers**Header versioning and media type versioning will misuse the headers.

**3.Caching**

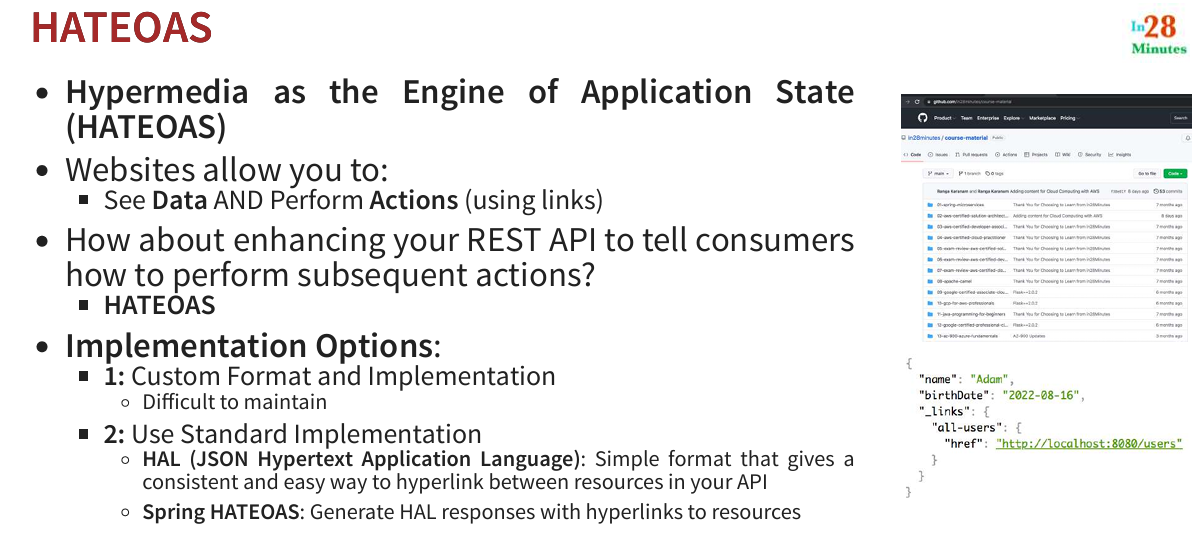
It is difficult for caching if you use header versioning and media type versioning.

**4.we can’t execute request for the above two in browser directly.**

**5.For API documentation it doesn’t support header versioning and media type versioning.**

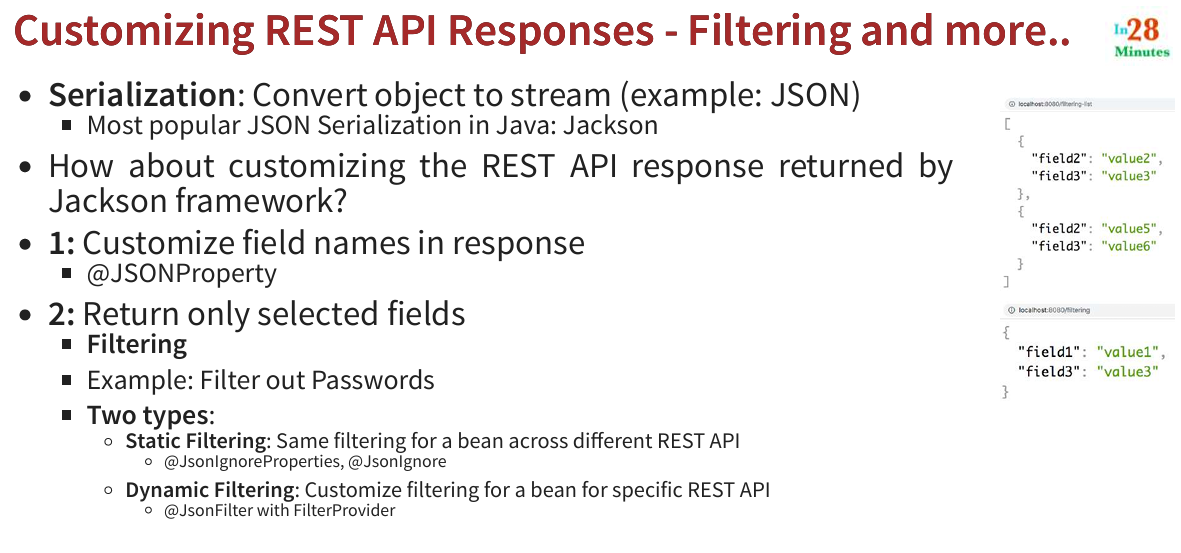
**May 5, 2024**

**HATEOS : H**ypertext **A**s **T**he **E**ngine **O**f **A**pplication **S**tate

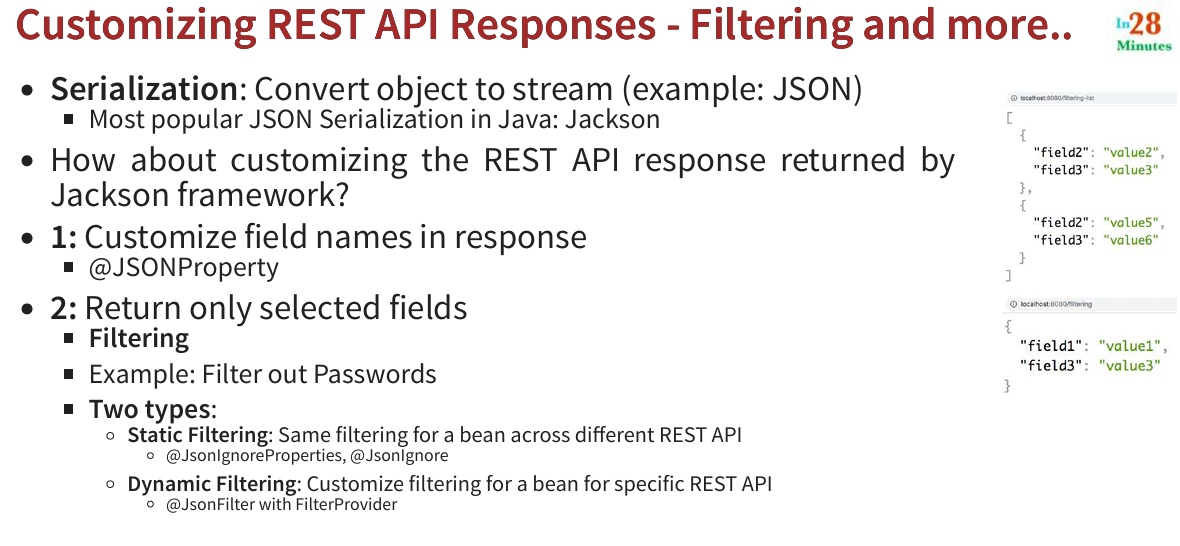


**Filtering**

**Types : static filtering and dynamic filtering**

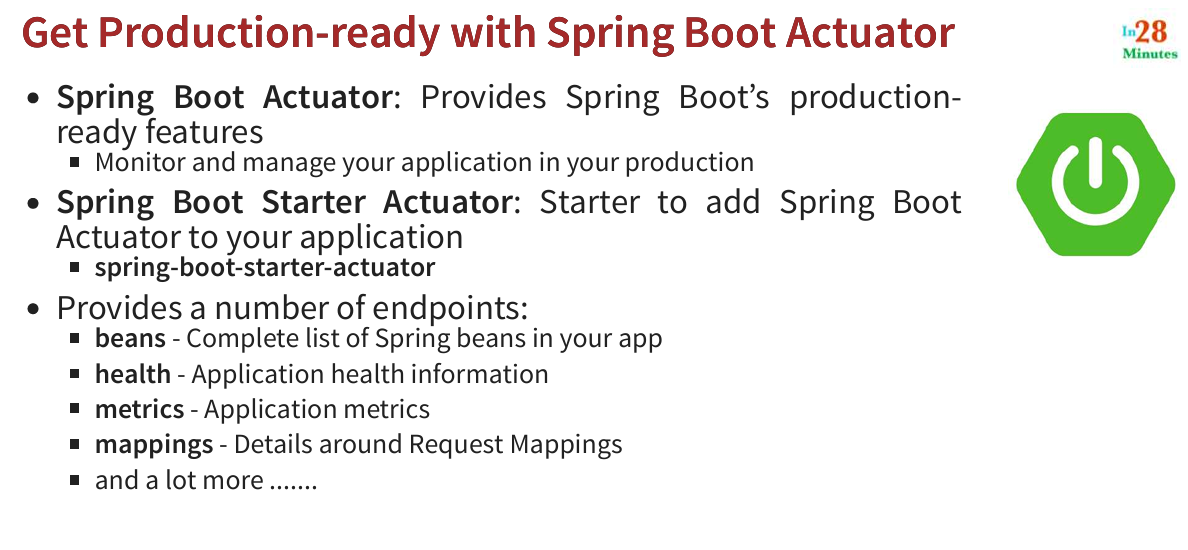


**Customization example:**

@JsonProperty(“user\_name”)  
private String name;  
  


Prefer using @JsonIgnore over @JsonIgnoreProperties because, at variable level we use @JsonIgnore but @JsonIgnoreProperties we will use it on class level and mention the variable names as well. If any variable name changes we need to change there as well. So, better go with @JsonIgnore.

**Spring-boot-Actuator**





**Learned how to use h2 database and console view of it. Learned how to run mysql container in docker to migrate from h2 to sql. Learned about spring security.**