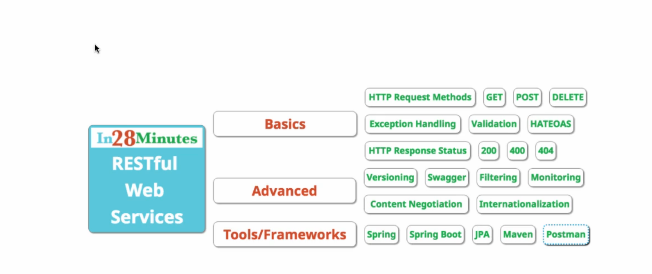
Spring Microservices:





What is web service ?

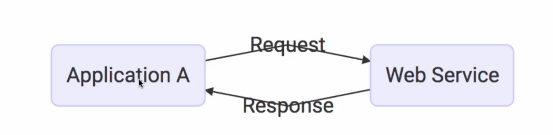
Software system designed to support interoperable machine to machine interaction over a network.

Designed to machine - machine interaction

Should be interoperable (platform dependent)

Should allow communication over network.

**How does data exchange between applications take place ?**



Request & response should be platform independent.

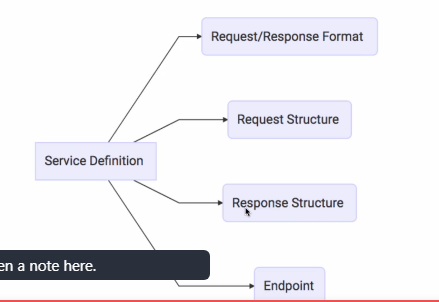
Data exchange formats -- 2 popular formats for request and response are XML & JSON (javascript object notation) formats.





**How does the application know the format of request & response ?**

Every web service offers a service definition.

****

**How can we make web services a platform independent ?**

By using data exchange formats like xml & JSON.

**Key Terminologies :**

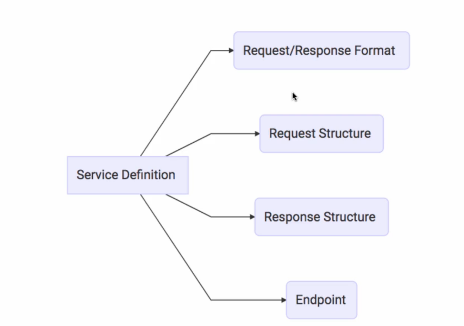
**Request & Response :** Request is input to a webservice. Response is output from a web service.

**Data exchange formats :** XML & JSON.

**Service provider & Service consumer :**

Service provider is the one which hosts the web service. Service consumer which is consuming the web service.

**Service Definition:** service definition is the contract b/w service provider & service consumer.



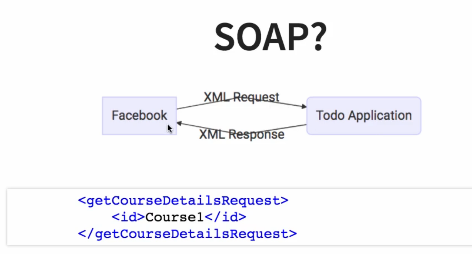
**End point** is the URL of the service.How can consumers call the service.

**Transport** : How the service is called over the web or Queue. HTTP / MQ.

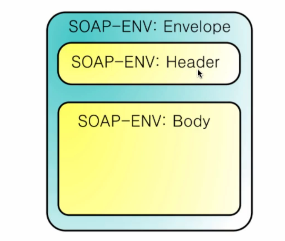
**Web Service groups:** SOAP & REST

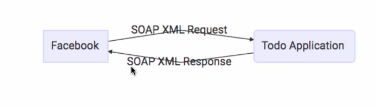
Soap & REST are not really comparable. Rest defines an architectural approach where as soap possess restrictions on the format of XML which is exchanged b/w provider & consumer.

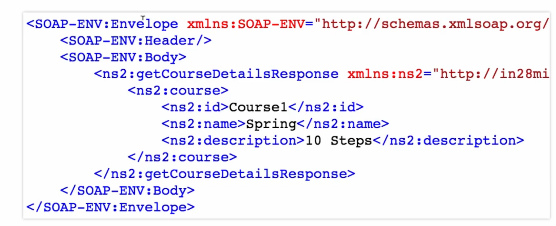
**Introduction to SOAP:**

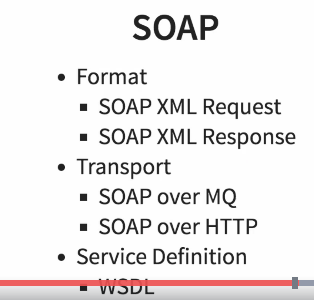


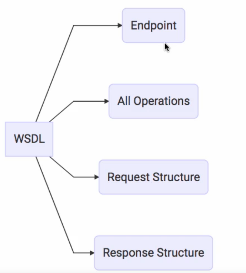
Soap defines a specific request and response structure.







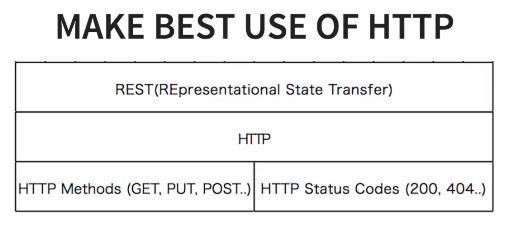
WSDL (web service definition language)



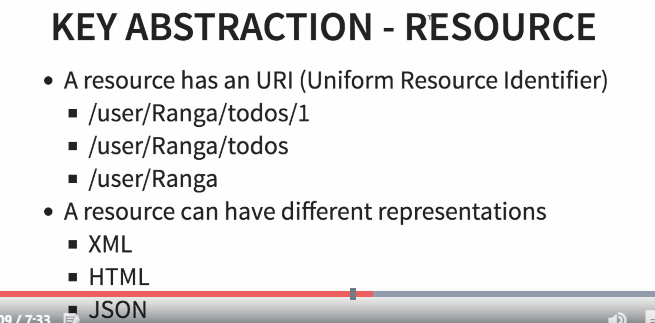
WSDL is service definition for SOAP service.

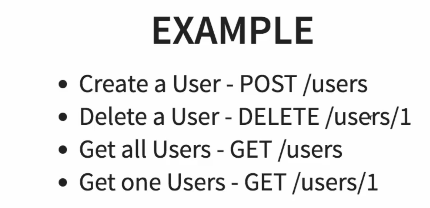


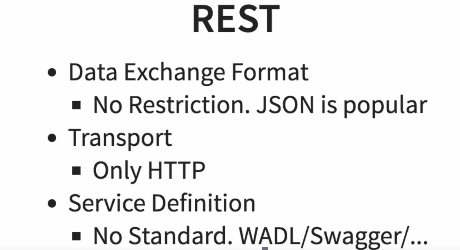
**Introduction to RESTful web services: REST - REpresentational State Transfer**

****

**HTTP - HyperText Transfer Protocol**

****

****

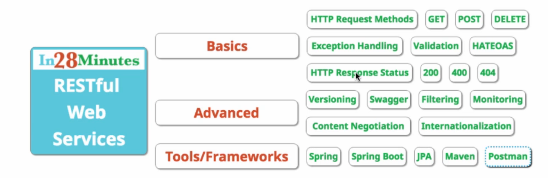
****

WADL --WEB Application Definition language (not popular)

Swagger is popular in defining restful web service.



**RESTful web services with Spring Boot:**

****

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

**#Restful web service**

**Social Media Application:**

-- Retrieve all users  **GET /users**

-- Create a user **POST /users**

-- Retrieve one user **GET /users/{id}**

-- Delete a user **DELETE /users/{id}**

-- Retrieve all posts for a user **GET /users/{id}/posts**

-- Create a post for a user **POST /users/{id}/posts**

-- Retrieve details of a post **POST /users/{id}/posts/{post-id}**

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

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

**What is dispatcher servlet?**

**Who is configuring the dispatcher servlet? --** Spring Boot Auto configuration

**What does the dispatcher servlet do?**

**How does HelloWorldBean object convert into json? --** Spring Boot Auto configuration & --jackson Object converters

**Who is configuring the dispatcher servlet?**

We have added the Spring Web as a dependency in the spring starter project. Web has a dependency on Spring Web MVC.so we got dispatcher servlet in our class path.

errorMVC configuration ,HTTP message converters, all these configurations fired on because of spring boot auto autoconfiguration.

Jackson objects converters responsible for converting object format to JSON, JSON to object format.

Dispatcher servlet is responsible for handling all the requests in spring application.Its following a pattern called frontcontroller.

DispatcherServlet maps requests to respective methods.

Once method returns response, @RESTController has response body.it converts response to JSON format.

**HATEOAS - Hypermedia as the engine of the application state**

To generate links in response

**Swagger:**documentation format for Restful web services.

Add dependencies of swagger2 & swagger-ui in pom.xml

Create swagger config class

**Swagger URL :** http://localhost:8080/swagger-ui/index.html#

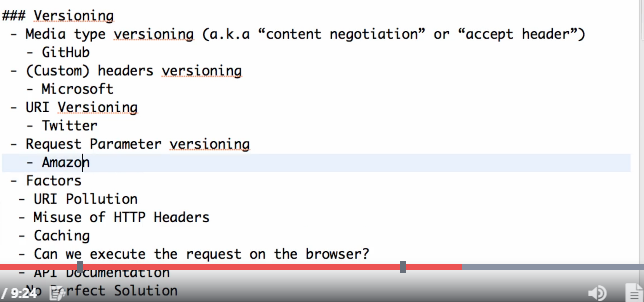
**Spring Boot Actuator :** to monitor all the services in a web application.

**Hal browser URL -- http://localhost:8080/browser/index.html#/**

**Filtering:**

Static filtering, dynamic filtering

**Versioning:**

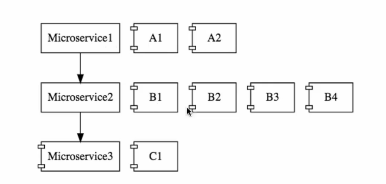


**Authentication:**include security dependency in pom.xml

**Restful to JPA:**

**Spring Micro services:**

****

****

**Challenges with microservices:**

1.Boundaries

2.configuration management(multiple micro services, multiple environments for each micro services and multiple instances for each micro service).

3.Dynamic scale up and scale down (Dynamic load balancing)

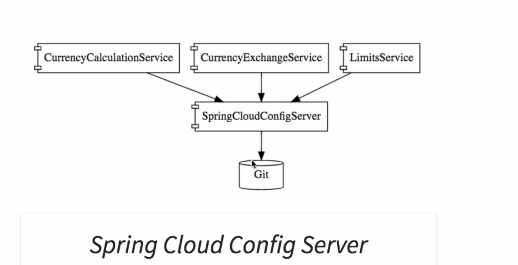
4.visibility(need centralized log which shows which microservice cause failure)

5.Pack of cards (one micro service failed, other micro service also failed based on that).

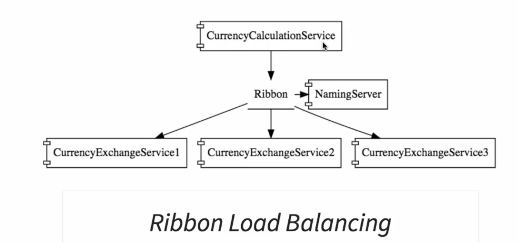
**Spring Cloud:**

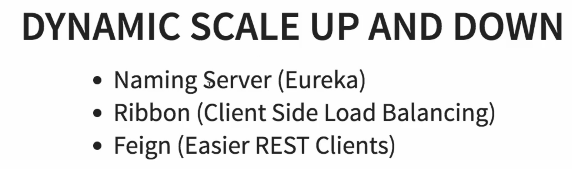
Spring cloud provides the solutions for challenges we faced in micro services.

1.configuration management -- All configurations maintained by Git repository.



2.Ribbon load balancing -- we will use ribbon for client side load balancing.





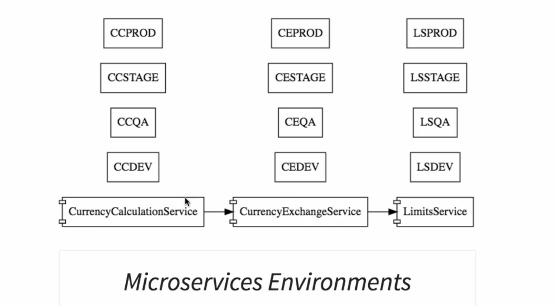


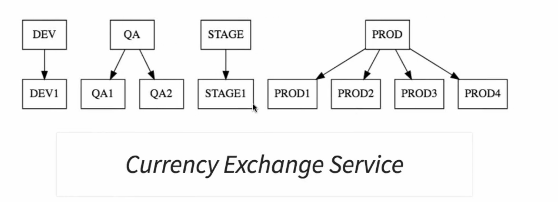
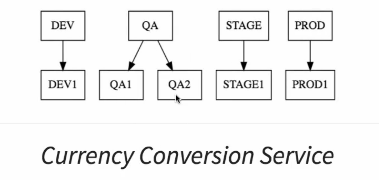


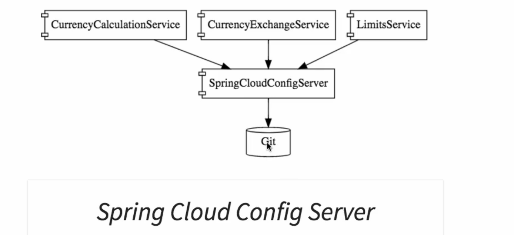
**Advantages of Micro services:**

1. It enables us to adopt new technologies and new processes easily.
2. Dynamic scaling -- same amount of load won’t be these throughout the year for Amazon like sites.
3. Faster Release cycles - smaller code easy to deploy.

**Microservices components:**

****

****

****

**1.setting up spring micro services**