WEB SERVICE - ARCHITETTURE WEB

Quando parliamo di ARCHITETTURE parliamo di Architetture Software.
Un' Architettura è un insieme di Applicazione Software

Esistono 2 tipologie di Architetture Web:

- **Architettura Tradizionale** (architettura con logica a spaghetti) [è un'architettura nella quale applicazioni scritte in linguaggi/framework differenti colloquiano tra di loro tramite interfacce applicative] e richiede la scrittura di tanto codice (linguaggio CORBA).
- **Architettura SOA** (**Service Oriented Architecture**) questa Architettura è un Architettura nella quale ogni Applicazione è pensata come un servizio, ovvero un applicazione che si espone al mondo esterno sottoforma di **Interfaccia Non Applicativa**, agnostica dalla tecnologia di implementazione grazie all'utilizzo di uno standard universale machine readable (comprensibile da tutte le applicazioni e tutte le macchine :XML/JSON)

Architettura SOA può essere di 2 tipologie:

1. **[Architettura Monolitica] =** Architettura in cui tutte le funzionalità vengono inserite all'interno della stessa applicazione.
   Questa può essere di 2 tipologie:
   - **Architettura SOAP** (XML come standard)
   - **Architettura REST** (JSON come standard)
2. **[Architettura A Microservizi] =** Architettura in cui ogni funzionalità viene scorporata in una diversa applicazione.
   Anche quest'architettura può essere di 2 tipologie:
   - **Architettura SOAP** (XML come standard)
   - **Architettura REST** (JSON come standard)

[SOAP] è un protocollo per lo scambio di messaggi tra componenti software.

[XML] = **verbose, cumbersome**.
E' molto prolisso, quindi significa che si deve scrivere molto.

[JSON] acronimo di **JavaScript Object Notation**
[JSON] = Light Weight, coincise
E' poco prolisso, quindi significa che si deve scrivere molto poco.

Se ROCCO LANDI ci fa il colloquio, di solito ci fa questa domanda:
Cos'é un WEB SERVICE? Come lo definisci?

Risposta:
In realtà ci sono tante definizioni di un WEB SERVICE.

Ci sono 2 tipi di WEB SERVICE:
- **WEB SERVICE CONSUMER**
- **WEB SERVICE PROVIDER**

[Un **WEB SERVICE PROVIDER** = E' un'Applicazione raggiungibile via Protocollo HTTP che si espone sottoforma di interfaccia non applicativa (XML/JSON).]
Quindi quest'Interfaccia Applicativa è indipendente dalla tecnologia usata per l'implementazione, e quindi consumabile da qualsiasi tecnologia utilizzata per l'implementazione dell'applicazione che la invoca.

Un **[WEB SERVICE PROVIDER]** può essere:
1. **SOAP WEB SERVICE PROVIDER**;
2. **RestFull WEB SERVICE PROVIDER.**

[SOAP WEB SERVICE PROVIDER = Un'Applicazione che si espone sotto forma di coppie composte da verbo HTTP + URI (stringa che rappresenta una Path) e che utilizza il formato XML per lo scambio di informazioni con il web service consumer (Applicazione che la interroga)]

L' Architettura SOA si compone di :
Un' Applicazione **WEB SERVICE CONSUMER** per consumare un'operazione esposta da un **WEB SERVICE PROVIDER**.
Deve fare una Richiesta tramite protocollo HTTP, identificando nella richiesta IP o DNS del server (sul quale risiede l'applicazione provider), la porta del server su cui viene eseguita l'applicazione.
Il Verbo HTTP dell'operazione che si intende consumare, è una URL Pattern corrispondente alla URI dell'operazione del web service provider.

I principali **[Verbi HTTP]** utilizzabili da un **WEB SERVICE PROVIDER** per l'esposizione delle operazioni sono :
1. **get** -> dovrebbe essere Utilizzato ***Per Operazioni di Lettura Pura***, cioè operazioni che non richiedono al consumer l'invio di informazioni nel corpo della richiesta.
2. **post** -> utilizzato sia ***Per Operazioni di Inserimento che per Operazioni di Aggiornamento*** (in questo caso si richiede al consumer l'invio di informazioni nel corpo della richiesta).
3. **put** -> utilizzato sia Per Operazioni di Inserimento che per operazioni di aggiornamento (in questo caso si richiede al consumer l'invio di informazioni nel corpo della richiesta).
4. **delete** -> da utilizzare per operazioni ...

Tecnologie Java per l'implementazione di un'Applicazione **WEB SERVICE PROVIDER**:

JAX-WS (API Java EE) -> ***SOAP WEB SERVICE PROVIDER*** (XML)

JAX-RS (API Java EE) -> per implementare ***RESTFULL WEB SERVICE PROVIDER***(XML/JSON)
Spring RestFul (Spring) -> ***RESTFULL WEB SERVICE PROVIDER***(xml/json) fornisce un insieme di API ed Annotation che consentono di implementare un RESTFULL WEB SERVICE

Se annotiamo una Classe Java con l'Annotation (prima della classe scriviamo l'annotation) [@RestController] la Classe diventa un **RESTFULL WEB SERVICE PROVIDER**

Inoltre Spring Restful fornisce tante Annotation per quanti sono i Verbi HTTP
- ***@GetMapping*** (per operazioni get)
- ***@PostMapping*** (per operazioni post)
- ***@PutMapping*** (per operazioni put)
- ***@DeleteMapping*** (per operazioni delete)

Se utilizziamo il modulo spring restful in un progetto Spring Boot, abbiamo un grande vantaggio, ovvero la produzione automatica degli XML/JSON di risposta verso il consumer

```
@RestController
//con annotation diventa sia springbean sia un web service provider
public class MyService{

@GetMapping("/Welcome") // "/Welcome" è la URI
public Object welcome(){

return Object;
}
}
```

Quando viene eseguita un'Applicazione Spring Boot, dietro le quinte viene eseguito un Oggetto all'interno dell'IoC container chiamato **jackson Object mapper** che riesce a convertire Oggetti java in un file JSON ...

Ogni file JSON deve essere composto da coppie proprietà valore inserite all'interno di parentesi graffe :

```
{
"message" : "hello"
}
```

[La **Dispatcher Servlet** verifica l'esistenza di un metodo inserito all'interno di un RestController che mappi il verbo della richiesta e l'URL Pattern della richiesta.]
Se non trova nessun Metodo restituisce la risposta Not Found, in caso positivo invoca il relativo...

---

https://restfulapi.net/idempotent-rest-apis/ ci fa vedere quale metodo è Idempotente e quale NON é Idempotente

Ogni Verbo HTTP Rest JAVA EE ha delle caratteristiche:
- **put --> Idempotent**
- **post --> Not Idempotent**

[Idempotent] = Ha origine dalla matematica.
[In informatica si dice che un'operazione è Idempotente, se l'operazione pur eseguita più volte (in maniera identica), non produce mai risultati diversi sul sistema, quindi produce sempre lo stesso risultato.]

Es. pratico per capire Idempotente :
Un pulsante Idempotente è come un interruttore della luce. Quando lo premi una volta, la luce si accende. Se lo premi di nuovo, la luce rimane accesa, non cambia nulla. Quindi, anche se lo premi tante volte, dopo la prima volta, non succede nulla di diverso. La luce rimane accesa.

Es. pratico per capire NON Idempotente
Un pulsante non idempotente è come un distributore automatico di caramelle. Quando premi il pulsante una volta, ottieni una caramella. Se lo premi di nuovo, ottieni un'altra caramella. Ogni volta che premi il pulsante, succede qualcosa di nuovo e ottieni un'altra caramella.

Spring implementa l'AOP (ASPECT ORIENTED PROGRAMMING)
[L'AOP] = metodologia di sviluppo che ha l'obiettivo di rilevare eventi che accadono a runtime.

Spring fornisce alcune Annotation che sono in grado di rilevare questi eventi che accadono a runtime, una di queste si chiama:

@RestControllerAdvice
[Quest'annotation usata sopra il nome della classe, **rende la classe una componente

applicativa speciale, in grado di porsi in ascolto su eventuali Eccezioni che si verifichino all'interno dei RestController di progetto.]**
[Possiamo anche abilitare tale classe ad eseguire delle azioni specifiche in base ad eventuali Eccezioni che si verifichino, come ad esempio la restituzione di JSON con Messaggi Custom]

[EntityModel] = API Spring Restful che consente di **implementare il pattern HATEOAS**

HATEOAS (HYPERMEDIA AS THE ENGINE OF APPLICATION STATE)
Secondo questo Pattern, [è preferibile restituire ai Web Service Consumer] (quelli che fanno le richieste), risposte che mostrino l'intero stato dell'applicazione Web Service Provider, ovvero restituire [risposte che contengano sia il contenuto informativo sulla base della richiesta specifica effettuata, sia un link che rimandi a tutte le altre possibili operazioni consumabili].
In pratica il Pattern HATEOAS consiglia di rendere navigabili le risposte fornite ai Web Service Consumer.

Lo stato di [un'applicazione Web Service Provider] = E' un'insieme delle operazioni Rest che un Web Service Provider espone.

SOAP Web Service = [riesce a restituire solo risposte XML]

Restful Web Service Provider = [riesce a restituire risposte sia sotto forma di XML, sia sotto forma di JSON]