**Service oriented Architecture (SOA)**

SOAP (simple Object Access Protocol): a description of an xml message protocol.

Benefits of SOA: reuse, integrate, deliver, distribute, standardize. (RIDDS)

Webservice protocol stack: service transport, xml messaging, service description, service discovery.

XML schema: describes an XML markup language. It defines which elements and attributes are used in a markup language.

Complex types describe how elements are organized and nested. Simple types are the primitive data types contained by elements and attributes.

An XML namespace provides a qualified name for an XML element or attribute.

Basic structure of a SOAP message: Envelope, header, body.

Document/Literal mode. In this mode, a SOAP Body element contains an XML document fragment, a well-formed XML element.

RPC/Literal mode. This mode enables SOAP messages to model calls to procedures or method calls with parameters and return values.

Messaging Exchange Patterns: One-way and Request/Response.

SOAP messages are delivered using HTTP Post.

WSDL: Web Service Description Language is an XML markup language used to described a Web service.

A WSDL document contains seven important elements: types, import, message, portType, operations, binding and service.

UDDI (Universal Description, Discovery and Integration) is a specification for creating a registry service.

Marshalling: is the process of transforming an object into a compatible version to be transmitted through the network.

POJO ----🡪 Marshalling 🡪JSON (String)

JSON (String) ->Marshalling -> POJO

JAX-B, JAX-WS, Jackson.

Jackson: JSON(String) <-> POJO: REST

JAX-B: XML <-> POJO: REST, SOAP

JAX-WS: WSDL <-> Java: SOAP

**REST (Representational State Transfer)**

Web services that follows the SOP standard and uses HTTP as its main protocol.

It supports JSON and XML for data transmission (representational).

Everything is considered a resource (state).

Objects are transmitted with the use of HTTP methods (transfer).

**HTTP methods**:

Post: used to create a new resource, most used.

Get: read only operations on resources.

Delete: removal of resources.

Put: used to update a resource.

Options: used to get supported operations on a resource

Head: used to send and get only head messages.

Patch: used to partially update a resource to consume less bandwidth, not idempotent methods.

**Rest Message** – Request: }

//ReponseEntity return response status

@RequestMapping(value=”/{id}”, method=RequestMethod.PUT,

Consumes=MediaType.Application\_Json\_Value,

Produces=MediaType.Application\_Json\_Value)

Public ResponseEntity<User> updateUser(@Pathvariable int id, @RequestBody User u) {

User user = service.updateUser(id, u);

If(user == null) {

Return new ResponseEntity<User> (HttpStatus.Conflict);

} else {

Return new ResponseEntity<User>(user, HttpStatus.OK);

}

HTTP Request:

* Verb: indicates the executing HTTP method
* URI: specifies the endpoint where resource is located.
* HTTP Version
* Request Header: META-DATA (information) of the Request such as: format supported by client, browser type.
* Request body: message content or resource representation.

**REST Message – Response**

HTTP Response:

Response code: 200 (ok), 403 (forbidden), 404 (Not Found), 500 (Internal Error).

HTTP Version

Response Header: Meta-Data for the response such as content length, content type, date.

Response body: resource representation.

**Spring REST**: Spring’s solution for RESTful web services.

* Includes as part of MVC.
* Change @Controller for @RestController.
* Once you do so: @ResponseBody is not needed anymore.
* You need to provide additional setup to multiple media types:

@RequestMapping(path=”/path”, produces={JSON, XML}) on the top of your controller.

**REST: Exception Handling** (Spring)

* Response Entity: new ResponseEntity<Customer> (HttpStatus.NOT\_FOUND)
* @ResponseStatus: on top of a custom exception.

@ResponseStatus(value=HttpStatus.NOT\_FOUND, reason = “Message”)

* @ExceptionHandler: on top of a method that is going to handle the exception, @ExceptionHandler (CustomException.class). It can also be done globally in a separate class which uses @ControllerAdvice.

**RESTFUL**: Structure, Cacheable, Stateless, Multiple layered system, Representation of resources, Client-server based architecture, Implementational freedom.

**Web Services: SOAP vs REST**

**The Tools**

SOAP: there are many SOAP engine tools available (Apache CXF)

REST: is Tool-less.

**The Exception Handling**

SOAP: has exception handling built-in, configurable in the body of the message (<fault> tag).

REST: behavior needs to be provided through Spring in order to be able to handle exceptions.

**The contract**

Soap: is contract based (WSDL).

REST: is not contract based.

**The Message**

SOAP: uses a standard structured message called Envelope, it only allows XML as its media type.

REST: uses the HTTP as part of the protocol, has a Request and a Response message, allows JSON (Jackson), XML (JAX-B), and other media types.

**The Atomicity**

Soap: has transactions built-in, configurable in the header of the message, this can make messages become atomic.

REST: does not perform atomic transactions, just represents state.

**The Protocol**

SOAP: can use HTTP, HTTPS, FTP, SMTP, and other protocols.

REST: can only use HTTP and HTTPS

**The Security**

Soap has security built-in, configurable in the header of the message, it allows double encryption, if HTTPS is being used.

REST: the only way to secure it is through HTTPS and authentication.

Remember: TECMAPS

EC2: Elastic Compute Cloud is a web-based service that allows businesses to run [application programs](https://searchsoftwarequality.techtarget.com/definition/application) in AWS public cloud.

To use EC2, a developer creates an AMI (Amazon Machine Image), containing an [operating system](https://whatis.techtarget.com/definition/operating-system-OS), application programs and configuration settings. The AMI is then uploaded to the S3 ( [Simple Storage Service](https://searchaws.techtarget.com/definition/Amazon-Simple-Storage-Service-Amazon-S3) ) and registered with EC2, creating an AMI identifier.

**Elastic Load Balancing** (ELB). ELB automatically distributes incoming application traffic and scales resources to meet traffic demands.

EBS: Elastic Block Service

DevOps is a set of practices that automates the processes between software development and IT teams, in order that they can build, test, and release software faster and more reliably.

**IAS:** infrastructure as service: RDS, S3.

PAS: platform as service: EC2

SAS: software as service

Continuous integration is the focus on merging and testing your code frequently.

Continuous delivery: when a merge to your remote repository results in a deliverable artifact, such as a WAR file.

Continuous deployment: when a merge to the remote repository results in your code being deployed to production.

Jenkins is an open source automation server written in Java. Jenkins helps to automate the non-human part of the software development process, with continuous integration and continuous delivery.

Jenkins pipeline consists of three stages: Build, Test, and Deploy.