TEK-UP Ecole Supérieure Privée Technologie & Ingénierie

Framework Spring (Spring Boot)

Wissem ELJAOUED
wissem.eljaoued@ensi-uma.tn

A.U. 2020-2021

Plan du cours

I. Spring - Introduction	W1- Project Creation
II. SB Components	W2- Domain + Repository layers
III. SB Persistence	W3- Service + Controller layers
IV. REST	W3- Controller REST layer
IV. Thymleaf	W4- View layer
IV. Spring Security	W5- Security layer

Ch. IV

REST

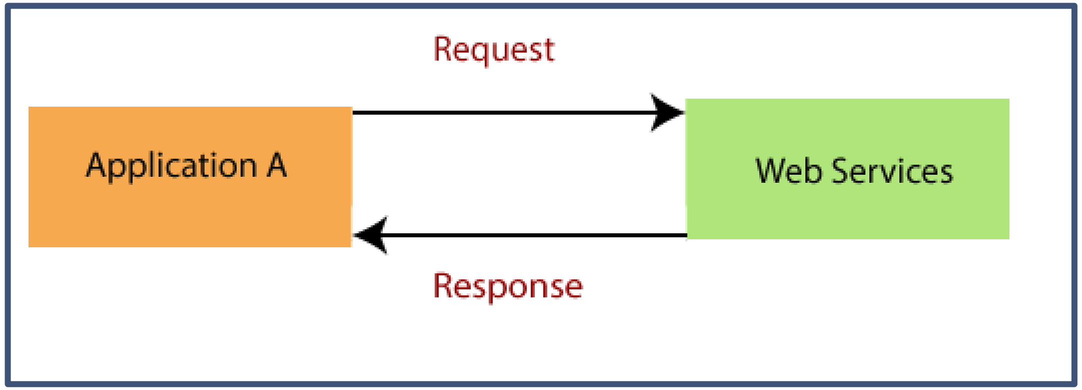
Ch. IV: REST

IV.1. Web service

- Web services are designed for application to application interaction.
- The web service must be accessible over the internet.
- Web services are used for reusing the code and connecting the existing program.
- Web services can be used to link data between two different platforms.

IV.1. Web service

- There are two popular formats for request and response **XML** and **JSON**.



```
[
  "employee":
  {
     "id": 00987
     "name":      "Jack",
     "salary":    20000,
  }
]
```

JSON

```
<getDetail>
<id>DataStructureCourse</id>
</getDetail>
```

XML

IV.1. Web service

- How does the Application A know the ***format*** of Request and Response?

- The answer to this question is ***Service Definition***. Every web service offers a service definition. Service definition specifies the following:

- ***Request/ Response format***: Defines the request format made by consumer and response format made by web service.
- ***Request Structure***: Defines the structure of the request made by the application.
- ***Response Structure***: Defines the structure of response returned by the web service.
- ***Endpoint***: Defines where the services are available.

Web service: Key concepts (1/2)

- *Request and Response:* Request is the input to a web service, and the response is the output from a web service.
- *Message Exchange Format:* It is the format of the request and response. There are two popular message exchange formats: ***XML*** and ***JSON***.
- *Service Provider or Server:* Service provider is one which ***hosts*** the web service.
- *Service Consumer or Client:* Service consumer is one who is using the web service.

Ch. IV: REST

- I.1. Web Service
- I.2. REST
- I.3. RESTful Spring Boot

Web service: Key concepts (2/2)

- ***Service Definition:*** Service definition is the contract between the service provider and service consumer. Service definition defines the format of request and response, request structure, response structure, and endpoint.

- ***Transport:*** Transport defines how a service is called. There is two popular way of calling a service: ***HTTP*** and Message Queue (***MQ***). By tying the URL of service, we can call the service over the internet. MQ communicates over the queue. The service requester puts the request in the queue. As soon as the service provider listens to the request. It takes the request, process the request, and create a response, and put the response back into MQ. The service requester gets the response from the queue. The communication happens over the queue.

IV.2. REST

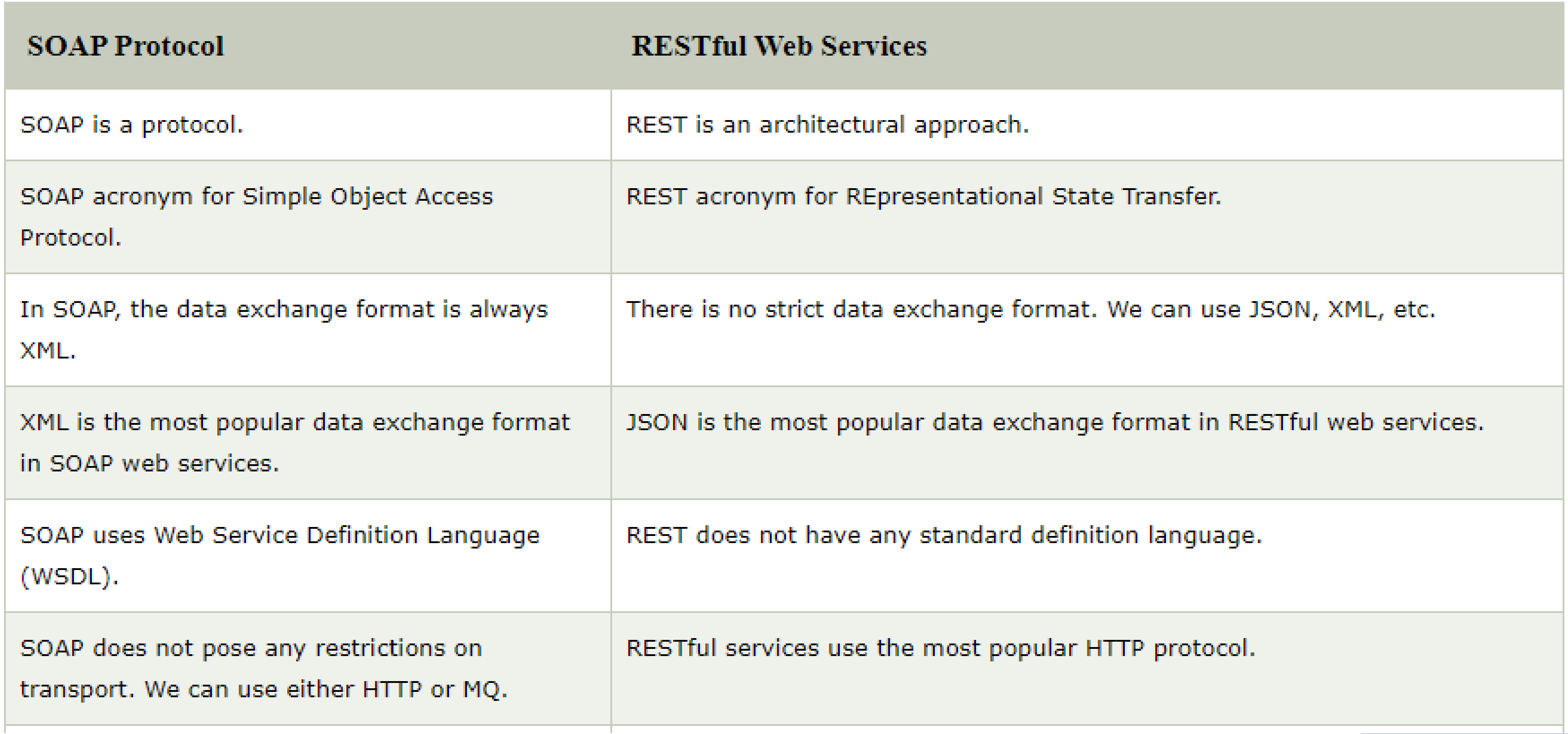
REpresentational State Transfer

It is an architectural style for building applications (Web, Intranet, Web Service). It is a set of conventions and good practices to be observed and not a technology in its own right. The REST architecture uses the original specifications of the HTTP protocol, rather than reinventing an overlay (as SOAP or XML-RPC do for example).

- It’s an architecture style
- It's not a standard
- It is an alternative to SOAP (Simple Object Access Protocol)(use only xml)

IV.2. REST

REST VS SOAP



SOAP Protocol	RESTful Web Services
SOAP is a protocol.	REST is an architectural approach.
SOAP acronym for Simple Object Access Protocol.	REST acronym for REpresentational State Transfer.
In SOAP, the data exchange format is always XML.	There is no strict data exchange format. We can use JSON, XML, etc.
XML is the most popular data exchange format in SOAP web services.	JSON is the most popular data exchange format in RESTful web services.
SOAP uses Web Service Definition Language (WSDL).	REST does not have any standard definition language.
SOAP does not pose any restrictions on transport. We can use either HTTP or MQ.	RESTful services use the most popular HTTP protocol.

IV.2. REST

- A ***RESTful Web Service*** is a function or method which can be called by sending an HTTP request to a URL, and the service returns the result as the response.

- RESTful web services try to define services using the different concepts that are already present in HTTP. REST is an architectural approach, not a protocol.

- We can build REST services with both XML and JSON. JSON is more popular format with REST. The key abstraction is a resource in REST. A resource can be anything. It can be accessed through a Uniform Resource Identifier (URI).

IV.2. REST

The important methods of HTTP are:

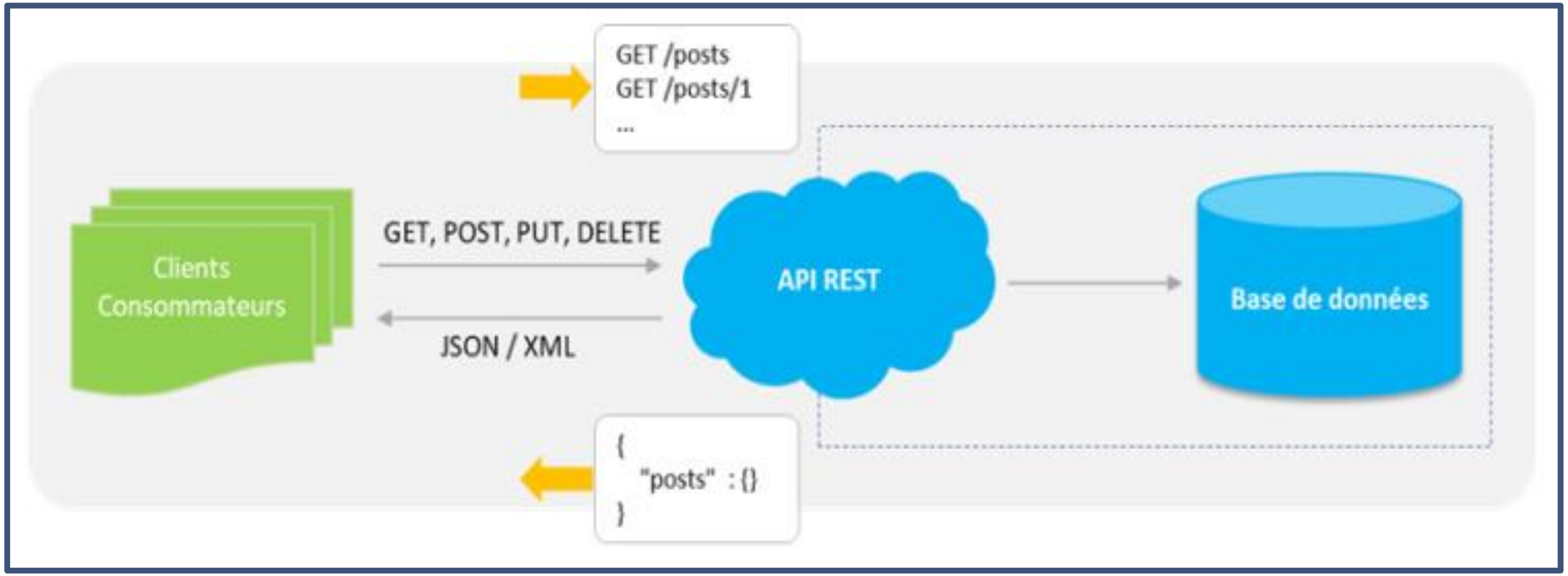
- *GET:* It reads a resource.
- *PUT:* It updates an existing resource.
- *POST:* It creates a new resource.
- *DELETE:* It deletes the resource.

IV.2. REST

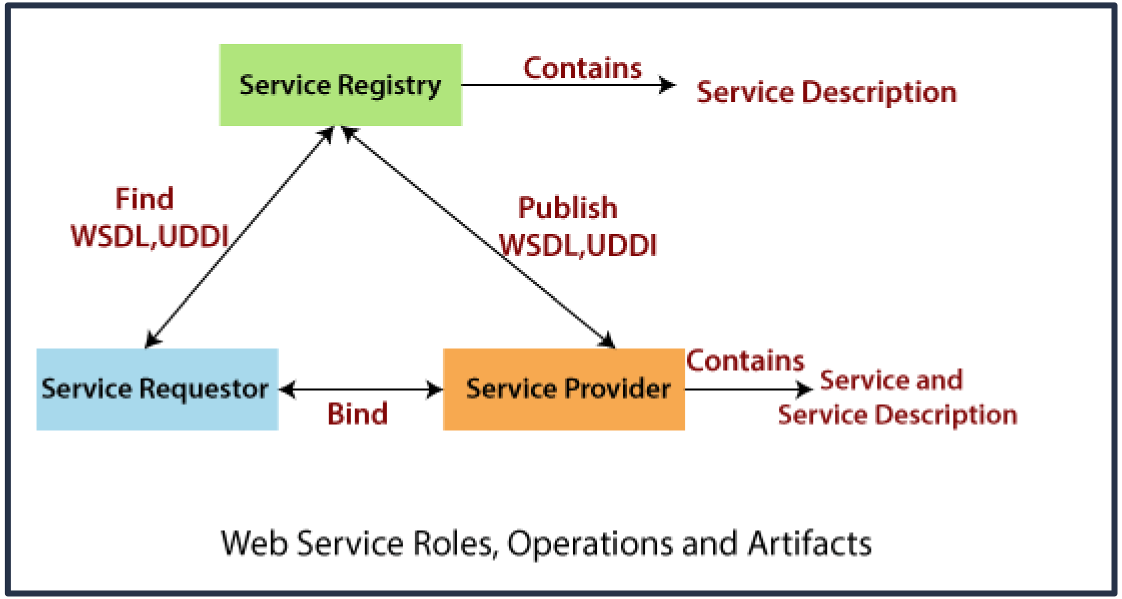
For example, if we want to perform the following actions in the social media application, we get the corresponding results

- *POST /users:* It creates a user.
- *GET /users/{id}:* It retrieves the detail of a user.
- *GET /users:* It retrieves the detail of all users.
- *DELETE /users:* It deletes all users.
- *DELETE /users/{id}:* It deletes a user.
- *GET /users/{id}/posts/post_id:* It retrieve the detail of a specific post.
- *POST / users/{id}/ posts:* It creates a post of the user.

IV.1. Rest


GET /posts
GET /posts/1
...
Clients Consommateurs
GET, POST, PUT, DELETE
JSON / XML
API REST
Base de données
{
"posts" : {}
}

IV.2. REST



Three **roles**:

- service **provider**
- service **requester**
- service **registry**

Three **operations**:

- **publish**
- **find**
- **bind**

IV.2. REST

Roles:

- **Service Provider:** From an architectural perspective, it is the platform that hosts the services.
- **Service Requestor:** Service requestor is the application that is looking for and invoking or initiating an interaction with a service. The browser plays the requester role, driven by a consumer or a program without a user interface.
- **Service Registry:** Service requestors find service and obtain binding information for services during development.

Actions:

- Publication of service descriptions (**Publish**)
- Finding of services descriptions (**Find**)
- Invoking of service based on service descriptions (**Bind**)

Ch. IV: REST

I.1. Web Service

I.2. REST

I.3. RESTful Spring Boot

Spring MVC + REST Annotations

@RestController: It can be considered as a combination of @Controller and @ResponseBody annotations. The @RestController annotation is itself annotated with the @ResponseBody annotation. It eliminates the need for annotating each method with @ResponseBody. It indicates that the data returned by each method will be written straight into the response body instead of rendering a template.

@RequestAttribute: It binds a method parameter to request attribute. It provides convenient access to the request attributes from a controller method. With the help of @RequestAttribute annotation, we can access objects that are populated on the server-side.

Spring MVC + REST Annotations

@GetMapping: It maps the HTTP GET requests on the specific handler method. It is used to create a web service endpoint that fetches It is used instead of using: @RequestMapping(method = RequestMethod.GET)

@PostMapping: It maps the HTTP POST requests on the specific handler method. It is used to create a web service endpoint that creates It is used instead of using: @RequestMapping(method = RequestMethod.POST)

@PutMapping: It maps the HTTP PUT requests on the specific handler method. It is used to create a web service endpoint that creates or updates It is used instead of using: @RequestMapping(method = RequestMethod.PUT)

@DeleteMapping: It maps the HTTP DELETE requests on the specific handler method. It is used to create a web service endpoint that deletes a resource. It is used instead of using: @RequestMapping(method = RequestMethod.DELETE)

Spring MVC + REST Annotations

@PatchMapping: It maps the HTTP PATCH requests on the specific handler method. It is used instead of using: @RequestMapping(method = RequestMethod.PATCH)

@RequestBody: It is used to bind HTTP request with an object in a method parameter. Internally it uses HTTP MessageConverters to convert the body of the request. When we annotate a method parameter with @RequestBody, the Spring framework binds the incoming HTTP request body to that parameter.

@ResponseBody: It binds the method return value to the response body. It tells the Spring Boot Framework to serialize a return an object into JSON and XML format.

@PathVariable: It is used to extract the values from the URI. It is most suitable for the RESTful web service, where the URL contains a path variable. We can define multiple @PathVariable in a method.

Spring MVC + REST Annotations

@RequestParam: It is used to extract the query parameters form the URL. It is also known as a query parameter. It is most suitable for web applications. It can specify default values if the query parameter is not present in the URL.

@RequestHeader: It is used to get the details about the HTTP request headers. We use this annotation as a method parameter. The optional elements of the annotation are name, required, value, defaultValue. For each detail in the header, we should specify separate annotations. We can use it multiple time in a method