**Name-Nabanita Sarkar**

**Employee ID-815944**

REST stands for **REpresentational State Transfer**. It is developed by **Roy Thomas Fielding**, who also developed HTTP. The main goal of RESTful web services is to make web services **more effective**. 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.

It does not define the standard message exchange format. 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)**. For example:

The resource has representations like XML, HTML, and JSON. The current state capture by representational resource. When we request a resource, we provide the representation of the resource. 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.

RESTful Service Constraints

* There must be a service producer and service consumer.
* The service is stateless.
* The service result must be cacheable.
* The interface is uniform and exposing resources.
* The service should assume a layered architecture.

Advantages of RESTful web services

* RESTful web services are **platform-independent**.
* It can be written in any programming language and can be executed on any platform.
* It provides different data format like **JSON, text, HTML,** and **XML**.
* It is fast in comparison to SOAP because there is no strict specification like SOAP.
* These are **reusable**.
* They are **language neutral**.

Difference between SOAP and RESTful Web Services

|  |  |
| --- | --- |
| **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. |
| SOAP web services are typical to implement. | RESTful services are easier to implement than SOAP. |
| SOAP web services use the JAX-WS API. | RESTful web services use the JAX-RS API. |
| SOAP protocol defines too many standards. | RESTful services do not emphasis on too many standards. |
| SOAP cannot use RESTful services because it is a protocol. | RESTful service can use SOAP web services because it is an architectural approach that can use any protocol like HTTP and SOAP. |
| SOAP reads cannot be cached. | REST reads can be cached. |

Rest Controller

The @RestController annotation is used to define the RESTful web services. It serves JSON, XML and custom response. Its syntax is shown below −

@RestController

public class ProductServiceController {

}

Request Mapping

The @RequestMapping annotation is used to define the Request URI to access the REST Endpoints. We can define Request method to consume and produce object. The default request method is GET.

@RequestMapping(value = "/products")

public ResponseEntity<Object> getProducts() { }

Request Body

The @RequestBody annotation is used to define the request body content type.

public ResponseEntity<Object> createProduct(@RequestBody Product product) {

}

Path Variable

The @PathVariable annotation is used to define the custom or dynamic request URI. The Path variable in request URI is defined as curly braces {} as shown below −

public ResponseEntity<Object> updateProduct(@PathVariable("id") String id) {

}

Request Parameter

The @RequestParam annotation is used to read the request parameters from the Request URL. By default, it is a required parameter. We can also set default value for request parameters as shown here −

public ResponseEntity<Object> getProduct(

@RequestParam(value = "name", required = false, defaultValue = "honey") String name) {

}

GET API

The default HTTP request method is GET. This method does not require any Request Body. You can send request parameters and path variables to define the custom or dynamic URL.

POST API

The HTTP POST request is used to create a resource. This method contains the Request Body. We can send request parameters and path variables to define the custom or dynamic URL.

PUT API

The HTTP PUT request is used to update the existing resource. This method contains a Request Body. We can send request parameters and path variables to define the custom or dynamic URL.

DELETE API

The HTTP Delete request is used to delete the existing resource. This method does not contain any Request Body. We can send request parameters and path variables to define the custom or dynamic URL.