@PathParam allows the rest service consumer to pass the input in service URI

@Path("{firstName}/{lastName}")

        @Produces(MediaType.*TEXT\_PLAIN*)

**public** String sayHello(@PathParam("firstName") String firstName, @PathParam("lastName") String lastName)

* 200 - success/OK
* 201 - CREATED - used in POST or PUT methods.
* 204-successful deletion
* 304 - NOT MODIFIED - used in conditional GET requests to reduce the bandwidth use of the network. Here, the body of the response sent should be empty.
* 400 - BAD REQUEST - This can be due to validation errors or missing input data.
* 401- UNAUTHORIZED - This is returned when there is no valid authentication credentials sent along with the request.
* 403 - FORBIDDEN - sent when the user does not have access (or is forbidden) to the resource.
* 404 - NOT FOUND - Resource method is not available.
* 500 - INTERNAL SERVER ERROR - server threw some exceptions while running the method.
* 502 - BAD GATEWAY - Server was not able to get the response from another

GET: This is used for fetching details from the server and is basically a read-only operation.

POST: This method is used for the creation of new resources on the server.

PUT: This method is used to update the old/existing resource on the server or to replace the resource.

DELETE: This method is used to delete the resource on the server.

PATCH: This is used for modifying the resource on the server.

OPTIONS: This fetches the list of supported options of resources present on the server.

| **SOAP** | **REST** |
| --- | --- |
| SOAP - Simple Object Access Protocol | REST - Representational State Transfer |
| SOAP is a protocol used to implement web services. | REST is an architectural design pattern for developing web services |
| SOAP cannot use REST as it is a protocol. | REST architecture can have SOAP protocol as part of the implementation. |
| SOAP specifies standards that are meant to be followed strictly. | REST defines standards but they need not be strictly followed. |
| SOAP client is more tightly coupled to the server which is similar to desktop applications having strict contracts. | The REST client is more flexible like a browser and does not depend on how the server is developed unless it follows the protocols required for establishing communication. |
| SOAP supports only XML transmission between the client and the server. | REST supports data of multiple formats like XML, JSON, MIME, Text, etc. |
| SOAP reads are not cacheable. | REST read requests can be cached. |
| SOAP uses service interfaces for exposing the resource logic. | REST uses URI to expose the resource logic. |
| SOAP is slower. | REST is faster. |

idempotent

The meaning of idempotent is that even after calling a single request multiple times, the outcome of the request should be the same.

GET, OPTIONS, TRACE, PUT , DELETE  and HEAD are idempotent

POST ,patch is not idempotent

methods are idempotent.

* Do you want to expose resource data or business logic?
  + SOAP is commonly used for exposing business logic and REST for exposing data.
* Does the client require a formal strict contract?
  + If yes, SOAP provides strict contracts by using WSDL. Hence, SOAP is preferred here.
* Does your service require support for multiple formats of data?
  + If yes, REST supports multiple data formats which is why it is preferred in this case.
* Does your service require AJAX call support?
  + If yes, REST can be used as it provides the XMLHttpRequest.
* Does your service require both synchronous and asynchronous requests?
  + SOAP has support for both sync/async operations.
  + REST only supports synchronous calls.
* Does your service require statelessness?
  + If yes, REST is suitable. If no, SOAP is preferred.
* Does your service require a high-security level?
  + If yes, SOAP is preferred. REST inherits the security property based on the underlying implementation of the protocol. Hence, it can’t be preferred at all times.
* Does your service require support for transactions?
  + If yes, SOAP is preferred as it is good in providing advanced support for transaction management.
* What is the bandwidth/resource required?
  + SOAP involves a lot of overhead while sending and receiving XML data, hence it consumes a lot of bandwidth.
  + REST makes use of less bandwidth for data transmission.
* Do you want services that are easy to develop, test, and maintain frequently?
  + REST is known for simplicity, hence it is preferred.

**How to make post as idempotent**

passing the unique Idempotency-key in every request made by the client

You can also set the expiry to your Idempotency-key to say 24hr

SolarWinds® ipMonitor® comes with two monitors, the HTTP monitor and HTTPS monitor, which can constantly check your HTTP and HTTPS endpoints to help you ensure they’re working properly.

<https://medium.com/javarevisited/five-api-performance-optimization-tricks-that-every-java-developer-must-know-75324ee1d244>