## Types of Web Services

On the conceptual level, a service is a software component provided through a network-accessible endpoint. The service consumer and provider use messages to exchange invocation request and response information in the form of self-containing documents that make very few assumptions about the technological capabilities of the receiver.

On a technical level, web services can be implemented in various ways. The two types of web services discussed in this section can be distinguished as “big” web services and “RESTful” web services.

### “Big” Web Services

In Java EE 6, JAX-WS provides the functionality for “big” web services, which are described in [Chapter 19, Building Web Services with JAX-WS](https://docs.oracle.com/javaee/6/tutorial/doc/bnayl.html). Big web services use XML messages that follow the Simple Object Access Protocol (SOAP) standard, an XML language defining a message architecture and message formats. Such systems often contain a machine-readable description of the operations offered by the service, written in the Web Services Description Language (WSDL), an XML language for defining interfaces syntactically.

The SOAP message format and the WSDL interface definition language have gained widespread adoption. Many development tools, such as NetBeans IDE, can reduce the complexity of developing web service applications.

A SOAP-based design must include the following elements.

* A formal contract must be established to describe the interface that the web service offers. WSDL can be used to describe the details of the contract, which may include messages, operations, bindings, and the location of the web service. You may also process SOAP messages in a JAX-WS service without publishing a WSDL.
* The architecture must address complex nonfunctional requirements. Many web service specifications address such requirements and establish a common vocabulary for them. Examples include transactions, security, addressing, trust, coordination, and so on.
* The architecture needs to handle asynchronous processing and invocation. In such cases, the infrastructure provided by standards, such as Web Services Reliable Messaging (WSRM), and APIs, such as JAX-WS, with their client-side asynchronous invocation support, can be leveraged out of the box.

### RESTful Web Services

In Java EE 6, JAX-RS provides the functionality for Representational State Transfer (RESTful) web services. REST is well suited for basic, ad hoc integration scenarios. RESTful web services, often better integrated with HTTP than SOAP-based services are, do not require XML messages or WSDL service–API definitions.

Project Jersey is the production-ready reference implementation for the JAX-RS specification. Jersey implements support for the annotations defined in the JAX-RS specification, making it easy for developers to build RESTful web services with Java and the Java Virtual Machine (JVM).

Because RESTful web services use existing well-known W3C and Internet Engineering Task Force (IETF) standards (HTTP, XML, URI, MIME) and have a lightweight infrastructure that allows services to be built with minimal tooling, developing RESTful web services is inexpensive and thus has a very low barrier for adoption. You can use a development tool such as NetBeans IDE to further reduce the complexity of developing RESTful web services.

A RESTful design may be appropriate when the following conditions are met.

* The web services are completely stateless. A good test is to consider whether the interaction can survive a restart of the server.
* A caching infrastructure can be leveraged for performance. If the data that the web service returns is not dynamically generated and can be cached, the caching infrastructure that web servers and other intermediaries inherently provide can be leveraged to improve performance. However, the developer must take care because such caches are limited to the HTTP GET method for most servers.
* The service producer and service consumer have a mutual understanding of the context and content being passed along. Because there is no formal way to describe the web services interface, both parties must agree out of band on the schemas that describe the data being exchanged and on ways to process it meaningfully. In the real world, most commercial applications that expose services as RESTful implementations also distribute so-called value-added toolkits that describe the interfaces to developers in popular programming languages.
* Bandwidth is particularly important and needs to be limited. REST is particularly useful for limited-profile devices, such as PDAs and mobile phones, for which the overhead of headers and additional layers of SOAP elements on the XML payload must be restricted.
* Web service delivery or aggregation into existing web sites can be enabled easily with a RESTful style. Developers can use such technologies as JAX-RS and Asynchronous JavaScript with XML (AJAX) and such toolkits as Direct Web Remoting (DWR) to consume the services in their web applications. Rather than starting from scratch, services can be exposed with XML and consumed by HTML pages without significantly refactoring the existing web site architecture. Existing developers will be more productive because they are adding to something they are already familiar with rather than having to start from scratch with new technology.

RESTful web services are discussed in [Chapter 20, Building RESTful Web Services with JAX-RS](https://docs.oracle.com/javaee/6/tutorial/doc/giepu.html). This chapter contains information about generating the skeleton of a RESTful web service using both NetBeans IDE and the Maven project management tool.

## Deciding Which Type of Web Service to Use

Basically, you would want to use RESTful web services for integration over the web and use big web services in enterprise application integration scenarios that have advanced quality of service (QoS) requirements.

* **JAX-WS**: addresses advanced QoS requirements commonly occurring in enterprise computing. When compared to JAX-RS, JAX-WS makes it easier to support the WS-\* set of protocols, which provide standards for security and reliability, among other things, and interoperate with other WS-\* conforming clients and servers.
* **JAX-RS**: makes it easier to write web applications that apply some or all the constraints of the REST style to induce desirable properties in the application, such as loose coupling (evolving the server is easier without breaking existing clients), scalability (start small and grow), and architectural simplicity (use off-the-shelf components, such as proxies or HTTP routers). You would choose to use JAX-RS for your web application because it is easier for many types of clients to consume RESTful web services while enabling the server side to evolve and scale. Clients can choose to consume some or all aspects of the service and mash it up with other web-based services.

**The most frequently asked RESTful Web services interview questions:**

Web services, a very well-known term when we talk about exchanging some sort of data between multiple applications or say software. Based on the client-server model, these services can be used by multiple software application written in various languages and it has an advantage of running on various platforms.

Likewise, REST, Representational State Transfer is also based on stateless client-server style architecture, which can be easily accessed over the network and is identified by URIs i.e. Uniform Resource Identifier.

The main aim of describing the definition of Web services as well as REST above is to help you relate to the term ‘RESTful web services’ because RESTful web services are defined as web services which use HTTP method and is based on the architecture of REST. It has useful features like high scalability and maintainability, the creation of APIs, etc.

In this article, you will find the collection of question and answer which will clear your basics and help develop the better understanding of the subject.

# Using the Spring @RequestMapping Annotation

### As often as Springs @RequestMapping annotation is, few recognize its versatility. Here, we see that on display when used to map Spring MVC controller methods.

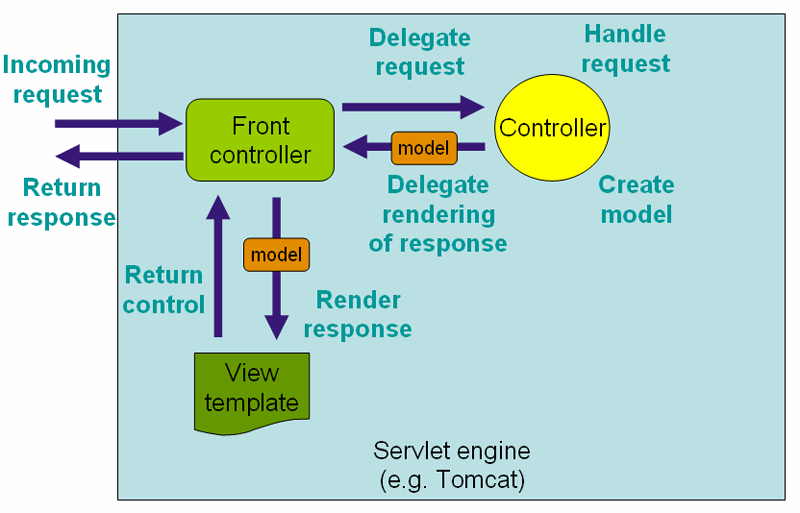
@RequestMapping is one of the most common annotation used in Spring Web applications. This annotation maps HTTP requests to handler methods of MVC and REST controllers.

In this post, you’ll see how versatile the @RequestMapping annotation is when used to map Spring MVC controller methods.

## Request Mapping Basics

In Spring MVC applications, the RequestDispatcher (Front Controller Below) servlet is responsible for routing incoming HTTP requests to handler methods of controllers.

When configuring Spring MVC, you need to specify the mappings between the requests and handler methods.

[](https://i2.wp.com/springframework.guru/wp-content/uploads/2017/09/mvc.png)To configure the mapping of web requests, you use the @RequestMapping annotation.

The @RequestMapping annotation can be applied to class-level and/or method-level in a controller.

The class-level annotation maps a specific request path or pattern onto a controller. You can then apply additional method-level annotations to make mappings more specific to handler methods.

Here is an example of the @RequestMapping annotation applied to both class and methods.

@RestController

@RequestMapping("/home")

public class IndexController {

@RequestMapping("/")

String get() {

//mapped to hostname:port/home/

return "Hello from get";

}

@RequestMapping("/index")

String index() {

//mapped to hostname:port/home/index/

return "Hello from index";

}

}

With the preceding code, requests to /home will be handled by get() while request to /home/index will be handled by index().

## @RequestMapping With Multiple URIs

You can have multiple request mappings for a method. For that add one @RequestMapping annotation with a list of values.

@RestController

@RequestMapping("/home")

public class IndexController {

@RequestMapping(value = {

"",

"/page",

"page\*",

"view/\*,\*\*/msg"

})

String indexMultipleMapping() {

return "Hello from index multiple mapping.";

}

}

As you can see in this code, @RequestMapping supports wildcards and ant-style paths. For the preceding code, all these URLs will be handled by indexMultipleMapping().

* localhost:8080/home
* localhost:8080/home/
* localhost:8080/home/page
* localhost:8080/home/pageabc
* localhost:8080/home/view/
* localhost:8080/home/view/view

## @RequestMapping With @RequestParam

The @RequestParam annotation is used with @RequestMapping to bind a web request parameter to the parameter of the handler method.

The @RequestParam annotation can be used with or without a value. The value specifies the request param that needs to be mapped to the handler method parameter, as shown in this code snippet.

@RestController

@RequestMapping("/home")

public class IndexController {

@RequestMapping(value = "/id")

String getIdByValue(@RequestParam("id") String personId) {

System.out.println("ID is " + personId);

return "Get ID from query string of URL with value element";

}

@RequestMapping(value = "/personId")

String getId(@RequestParam String personId) {

System.out.println("ID is " + personId);

return "Get ID from query string of URL without value element";

}

}

In Line 6 of this code, the request param id will be mapped to the personId parameter personId of thegetIdByValue() handler method.

The value element of @RequestParam can be omitted if the request param and handler method parameter names are same, as shown in Line 11.

The required element of @RequestParam defines whether the parameter value is required or not.

@RestController

@RequestMapping("/home")

public class IndexController {

@RequestMapping(value = "/name")

String getName(@RequestParam(value = "person", required = false) String personName) {

return "Required element of request param";

}

}

In this code snippet, as the required element is specified as false, the getName() handler method will be called for both of these URLs:

* /home/name?person=xyz
* /home/name

The default value of the @RequestParam is used to provide a default value when the request param is not provided or is empty.

@RestController

@RequestMapping("/home")

public class IndexController {

@RequestMapping(value = "/name")

String getName(@RequestParam(value = "person", defaultValue = "John") String personName) {

return "Required element of request param";

}

}

In this code, if the person request param is empty in a request, the getName() handler method will receive the default value John as its parameter.

## Using @RequestMapping With HTTP Methods

The Spring MVC @RequestMapping annotation is capable of handling HTTP request methods, such as GET, PUT, POST, DELETE, and PATCH.

By default, all requests are assumed to be of HTTP GET type.

In order to define a request mapping with a specific HTTP method, you need to declare the HTTP method in@RequestMapping using the method element as follows.

@RestController

@RequestMapping("/home")

public class IndexController {

@RequestMapping(method = RequestMethod.GET)

String get() {

return "Hello from get";

}

@RequestMapping(method = RequestMethod.DELETE)

String delete() {

return "Hello from delete";

}

@RequestMapping(method = RequestMethod.POST)

String post() {

return "Hello from post";

}

@RequestMapping(method = RequestMethod.PUT)

String put() {

return "Hello from put";

}

@RequestMapping(method = RequestMethod.PATCH)

String patch() {

return "Hello from patch";

}

}

In the code snippet above, the method element of the @RequestMapping annotations indicates the HTTP method type of the HTTP request.

All the handler methods will handle requests coming to the same URL ( /home), but will depend on the HTTP method being used.

For example, a POST request to /home will be handled by the post() method. While a DELETE request to/home will be handled by the delete() method.

You can see how Spring MVC will map the other methods using this same logic.

## Using @RequestMapping With Producible and Consumable

The request mapping types can be narrowed down using the produces and consumes elements of the@RequestMapping annotation.

In order to produce the object in the requested media type, you use the produces element of @RequestMapping in combination with the @ResponseBody annotation.

You can also consume the object with the requested media type using the consumes element of@RequestMapping in combination with the @RequestBody annotation.

The code to use producible and consumable with @RequestMapping is this.

@RestController

@RequestMapping("/home")

public class IndexController {

@RequestMapping(value = "/prod", produces = {

"application/JSON"

})

@ResponseBody

String getProduces() {

return "Produces attribute";

}

@RequestMapping(value = "/cons", consumes = {

"application/JSON",

"application/XML"

})

String getConsumes() {

return "Consumes attribute";

}

}

In this code, the getProduces() handler method produces a JSON response. The getConsumes() handler method consumes JSON as well as XML present in requests.

## @RequestMapping With Headers

The @RequestMapping annotation provides a header element to narrow down the request mapping based on headers present in the request.

You can specify the header element as myHeader = myValue.

@RestController

@RequestMapping("/home")

public class IndexController {

@RequestMapping(value = "/head", headers = {

"content-type=text/plain"

})

String post() {

return "Mapping applied along with headers";

}

}

In the above code snippet, the headers attribute of the @RequestMapping annotation narrows down the mapping to the post() method. With this, the post() method will handle requests to /home/head whose content-typeheader specifies plain text as the value.

You can also indicate multiple header values like this:

@RestController

@RequestMapping("/home")

public class IndexController {

@RequestMapping(value = "/head", headers = {

"content-type=text/plain",

"content-type=text/html"

}) String post() {

return "Mapping applied along with headers";

}

}

Here it implies that both text/plain as well as text/html are accepted by the post() handler method.

## @RequestMapping With Request Parameters

The params element of the @RequestMapping annotation further helps to narrow down request mapping. Using the params element, you can have multiple handler methods handling requests to the same URL, but with different parameters.

You can define params as myParams = myValue. You can also use the negation operator to specify that a particular parameter value is not supported in the request.

@RestController

@RequestMapping("/home")

public class IndexController {

@RequestMapping(value = "/fetch", params = {

"personId=10"

})

String getParams(@RequestParam("personId") String id) {

return "Fetched parameter using params attribute = " + id;

}

@RequestMapping(value = "/fetch", params = {

"personId=20"

})

String getParamsDifferent(@RequestParam("personId") String id) {

return "Fetched parameter using params attribute = " + id;

}

}

In this code snippet, both the getParams() and getParamsDifferent() methods will handle requests coming to the same URL (/home/fetch) but will execute depending on the params element.

For example, when the URL is /home/fetch?id=10, the getParams() handler method will be executed with the id value 10.. For the URL, localhost:8080/home/fetch?personId=20, the getParamsDifferent() handler method gets executed with the id value 20.

## Using @RequestMapping With Dynamic URIs

The @RequestMapping annotation is used in combination with the @PathVaraible annotation to handle dynamic URIs. In this use case, the URI values can act as the parameter of the handler methods in the controller. You can also use regular expressions to only accept the dynamic URI values that match the regular expression.

@RestController

@RequestMapping("/home")

public class IndexController {

@RequestMapping(value = "/fetch/{id}", method = RequestMethod.GET)

String getDynamicUriValue(@PathVariable String id) {

System.out.println("ID is " + id);

return "Dynamic URI parameter fetched";

}

@RequestMapping(value = "/fetch/{id:[a-z]+}/{name}", method = RequestMethod.GET)

String getDynamicUriValueRegex(@PathVariable("name") String name) {

System.out.println("Name is " + name);

return "Dynamic URI parameter fetched using regex";

}

}

In this code, the method getDynamicUriValue() will execute for a request to localhost:8080/home/fetch/10. Also, the id parameter of the getDynamicUriValue() handler method will be populated with the value 10 dynamically.

The method getDynamicUriValueRegex() will execute for a request to localhost:8080/home/fetch/category/shirt. However, an exception will be thrown for a request to /home/fetch/10/shirt as it does not match the regular expression.

@PathVariable works differently from @RequestParam. You use @PathVariable to obtain the values of the query parameters from the URI. On the other hand, you use @RequestParam to obtain the parameter values from the URI template.

## The @RequestMapping Default Handler Method

In the controller class, you can have default handler method that gets executed when there is a request for a default URI.

Here is an example of a default handler method.

@RestController

@RequestMapping("/home")

public class IndexController {

@RequestMapping()

String

default () {

return "This is a default method for the class";

}

}

In this code, a request to /home will be handled by the default() method as the annotation does not specify any value.

## @RequestMapping Shortcuts

[Spring 4.3](https://spring.io/blog/2016/06/10/spring-framework-4-3-goes-ga) introduced method-level variants, also known as composed annotations of @RequestMapping. The composed annotations better express the semantics of the annotated methods. They act as wrapper to@RequestMapping and have become the standard ways of defining the endpoints.

For example, @GetMapping is a composed annotation that acts as a shortcut for @RequestMapping(method =RequestMethod.GET).  
The method level variants are:

* @GetMapping
* @PostMapping
* @PutMapping
* @DeleteMapping
* @PatchMapping

The following code shows using the composed annotations.

@RestController

@RequestMapping("/home")

public class IndexController {

@GetMapping("/person")

public @ResponseBody ResponseEntity < String > getPerson() {

return new ResponseEntity < String > ("Response from GET", HttpStatus.OK);

}

@GetMapping("/person/{id}")

public @ResponseBody ResponseEntity < String > getPersonById(@PathVariable String id) {

return new ResponseEntity < String > ("Response from GET with id " + id, HttpStatus.OK);

}

@PostMapping("/person")

public @ResponseBody ResponseEntity < String > postPerson() {

return new ResponseEntity < String > ("Response from POST method", HttpStatus.OK);

}

@PutMapping("/person")

public @ResponseBody ResponseEntity < String > putPerson() {

return new ResponseEntity < String > ("Response from PUT method", HttpStatus.OK);

}

@DeleteMapping("/person")

public @ResponseBody ResponseEntity < String > deletePerson() {

return new ResponseEntity < String > ("Response from DELETE method", HttpStatus.OK);

}

@PatchMapping("/person")

public @ResponseBody ResponseEntity < String > patchPerson() {

return new ResponseEntity < String > ("Response from PATCH method", HttpStatus.OK);

}

}

In this code, each of the handler methods are annotated with the composed variants of @RequestMapping. Although each variant can be interchangeably used with @RequestMapping with the method attribute, it’s considered a best practice to use the composed variant —primarily because the composed annotations reduce the configuration metadata on the application side and the code is more readable.

## @RequestMapping Conclusion

As you can see in this post, the @RequestMapping annotation is very versatile. You can use this annotation to configure Spring MVC to handle a variety of use cases. It can be used to configure traditional web page requests as well as RESTFul web services in Spring MVC.

**Also read:**

* [**Top Web Services Interview Questions**](http://www.softwaretestinghelp.com/web-services-interview-questions/)
* [**Top SoapUI interview questions**](http://www.softwaretestinghelp.com/soapui-interview-questions-and-answers/)

## RESTful web services Question and Answers

Let’s start.

**Q #1) What is your understanding of what are RESTful web services?**

Just like SOAP (Simple Object Access Protocol), which is used to develop web services by XML method, RESTful web services use web protocol i.e. HTTP protocol method. They have the feature like scalability, maintainability, help multiple application communication built on various programming languages etc.

RESTful web service implementation defines the method of accessing various resources which are required by the client and he has sent the request to the server through the web browser. The important aspects of this implementation include:

* Resources
* Request Headers
* Request Body
* Response Body
* Status codes

**Q #2) Name the protocol which is used by RESTful web services.**

RESTful web services use a famous web protocol i.e. HTTP protocol. This serves as a medium of data communication between client and server. HTTP standard methods are used to access resources in RESTful web service architecture.

**Q #3) Explain the term ‘Addressing’ with respect to RESTful WEB service.**

Just like we require address with postal code to reach any person, in the same way, ‘Addressing’ locates resources that are present on the server for the purpose of hosting web services. This is usually done with URI i.e. Unified Resource Identifier.

**Q #4) Enlist features of RESTful web services.**

Every RESTful web services should have following features and characteristics that are enlisted below:

* Based on the Client Server representation.
* Use of HTTP protocol for performing functions like fetching data from the web service, retrieving resources, execution of any query, etc.
* The communication between the server and client is performed through the medium known as ‘messaging’.
* Addressing of resources available on the server through URIs.
* Based on the concept of statelessness where every client request and the response is independent of the other with complete assurance of providing required information.
* Uses the concept of caching.
* Works on Uniform interface.

**Q #5) Explain messaging technique.**

Messages are the mode of exchanging data for any type of communication to take place. In the same way, HTTP protocol plays the role of message communication between the client and server through HTTP Request and Response methods. HTTP request is sent by the client who contains information about the data and in turn, receives HTTP Response from the server.

Messages are the collection of information about the data i.e. Metadata.

**Q #6) What are the core components of HTTP request and HTTP response?**

The core components that come under HTTP Request are:

* **Verb:** Includes methods like GET, PUT, POST, etc.
* Uniform Resource Identifier for identifying the resources available on the server.
* HTTP Version for specifying the HTTP version.
* HTTP Request header for containing the information about the data.
* HTTP Request body that contains the representation of the resources in use.

**The core components that come under HTTP Response are:**

* **Request Code:** This contains various codes which determine the status of the server response.
* HTTP Version for specifying the HTTP version.
* HTTP Response header for containing the information about the data.
* HTTP Response body that contains the representation of the resources in use.

**Q #7) Explain the term ‘Statelessness’ with respect to RESTful WEB service.**

In REST, ST itself defines State Transfer and Statelessness means complete isolation. This means, the state of the client’s application is never stored on the server and is passed on. In this process, the clients send all the information that is required for the server to fulfill the HTTP request that has been sent. Thus every client request and the response is independent of the other with complete assurance of providing required information.

Every client passes a ‘session identifier’ which also acts as an identifier for each session.

**Q #8) Enlist advantages and disadvantages of ‘Statelessness’.**

In the above question, we have understood the meaning of statelessness with respect to the client-server communication. Now, let us see some of its advantages and disadvantages.

**Advantages:**

* Every method required for communication is identified as an independent method i.e. there are no dependencies to other methods.
* Any previous communication with the client and server is not maintained and thus the whole process is very much simplified.
* If any information or metadata used earlier in required in another method, then the client sends again that information with HTTP request.
* HTTP protocol and REST web service, both shares the feature of statelessness.

**Disadvantages:**

* In every HTTP request from the client, the availability of some information regarding the client state is required by the web service.

**Q #9) Enlist some important constraints for RESTful web services.**

Every constraint has positive as well as negative impacts and to produce an overall architecture, there should be the balance between both. Below mentioned are some important constraints for RESTful web service:

* There should be separate concerns for each server and client which will help to maintain the modularity within the application. This will also reduce the complexity and increase the scalability.
* The client-server communication should be stateless, which means no previous information is used and the complete execution is done in isolation. In cases of failure, it also helps the client to recover.
* In client-server communication, the HTTP response should be cacheable so that when required cached copy can be used which in turn enhances the scalability and performance of the server.
* The fourth constraint is the uniform interface which allows client-server interaction to be easily understood. This constraint is further divided into four sub-constraints as:
  + Resource Identification
  + Resource manipulation
  + Each message is easily understood and is self-descriptive.
  + Hypermedia, which is defined as the text with hyperlinks and when clicked it moves to another application state.
* Client-server communication should be done on a layered system and thus the client should only have knowledge about the intermediate level with which communication is being done,

**Q #10) What is a ‘Resource’?**

Just like the ‘Object’ instance, we have learned in object orient programming Language, in the same way, ‘Resource’ is defined as an object of a type which can be an image, HTML file, text data, and any type of dynamic data. There are varieties of representation formats available in order to represent a resource.

**Some most common are enlisted below:**

* JSON
* YAML
* XML
* HTML

**Q #11) Why proper representation of Resource is required?**

Representation is very important because it determines the easy identification of resources. With proper representations of resource in the proper format, allows the client to easily understand the format.

**Q #12) Enlist some important points that should be kept in mind while designing Resources representation for RESTful web services.**

As there are no restrictions on the format in which the resource representation is done but just that the main requirement is the format of the representation should be as per the client requirement. A good resource representation is designed by considering the following main points:

* The resource representation format should be easily understood by the client and server.
* The representation should be complete regardless of its format structure, which may be complex or simple.
* In the case of the link of the resources to other resources, such cases should also be considered and handled.

**Q #13) What is Caching?**

Caching is the process in which server response is stored so that a cached copy can be used when required and there is no need of generating the same response again. This process not only reduces the server load but in turn increase the scalability and performance of the server. Only the client is able to cache the response and that too for a limited period of time.

Mentioned below are the header of the resources and their brief description so that they can be identified for the caching process:

* Time and Date of resource creation
* Time and date of resource modification that usually stores the last detail.
* Cache control header
* Time and date at which the cached resource will expire.
* The age which determines the time from when the resource has been fetched.

**Q #14) Explain Cache-control header.**

A standard Cache control header can help in attaining cache ability. Enlisted below is the brief description of various cache control header:

* **Public:** Resources that are marked as the public can be cached by any intermediate components between the client and server.
* **Private:** Resources that are marked as private can only be cached by the client.
* No cache means that resource cannot be cached and thus the whole process is stopped.

**Q #15) What are the best practices that are to be followed while designing RESTful web services?**

To design a secure RESTful web service, there are some best practices or say points that should be considered. These are explained as follows:

* Every input on the server should be validated.
* Input should be well formed.
* Never pass any sensitive data through URL.
* For any session, the user should be authenticated.
* Only HTTP error messages should be used for indicating any fault.
* Use message format that is easily understood and is required by the client.
* Unified Resource Identifier should be descriptive and easily understood.

**Q #16) What is Payload?**

The request data which is present in the body part of every HTTP message is referred as ‘Payload’.  In Restful web service, the payload can only be passed to the recipient through POST method.

There is no limit of sending data as payload through POST method but the only concern is that more data with consuming more time and bandwidth. This may consume much of user’s time also.

**Q #17) Enlist some of the HTTP methods with description.**

Mentioned below is the list of HTTP methods with their descriptions:

* **GET:** This is a read only operation which fetches the list of users on the server.
* **PUT:** This operation is used for the creation of any new resource on the server.
* **POST:** This operation is used for updating an old resource or for creating a new resource.
* **DELETE:** As the name suggests, this operation is used for deleting any resource on the server.
* **OPTIONS:** This operation fetches the list of any supported options of resources that are available on the server.

**Q #18) What is the difference between PUT method and POST method?**

The major difference between the PUT and POST method is that the result generated with PUT method is always same no matter how many times the operation is performed. On the other hand, the result generated by POST operation is always different every time.

**Q #19) What is your understanding about JAX-RS?**

JAX-RS is defined as the Java API for RESTful web service. Among multiple libraries and framework, this is considered as the most suitable Java programming language based API which supports RESTful web service.

**Some of the implementations of JAX-RS are:**

* Jersey
* RESTEasy
* Apache CFX
* Play

Among these, Jersey is the most popular framework.

**Q #20) What are HTTP status codes? Enlist few with meaning.**

HTTP status codes basically are the representation of the status of the task that has been performed on the server, with the mode of some codes. Every code has their own meaning.

**Some of the HTTP status codes with their meaning are as follows:**

* **Code 200:** This indicates success.
* **Code 201:** This indicates resource has been successfully created.
* **Code 204:** This indicates that there is no content in the response body.
* **Code 404:** This indicates that there is no method available.

There are few more such codes that indicate the status.

### Conclusion:

This article will help you prepare for the RESTful web services interview and help you to understand the concept in the simple and easy way. I have tried to cover all the areas which are very necessary for having the complete knowledge about RESTful Web services.

Just remember, it may be possible that you are not able to answer all questions in the interview but whatever you answer should be accurate. Your basic concept should be strong and your confidence level should be high.

**What does REST stand for?**(answer)  
REST stands for REpresentational State Transfer, which uses HTTP protocol to send data from client to server e.g. a book in the server can be delivered to the client using JSON or XML.  
  
  
**What is a resource?**(answer)  
A resource is how data is represented in REST architecture. By exposing entities as the resource it allows a client to read, write, modify, and create resources using HTTP methods e.g. [GET](http://javarevisited.blogspot.sg/2012/03/get-post-method-in-http-and-https.html), [POST](http://www.java67.com/2014/08/difference-between-post-and-get-request.html), [PUT](http://www.java67.com/2016/09/when-to-use-put-or-post-in-restful-web-services.html), DELETE etc.  
  
  
**What are safe REST operations?**(answer)  
REST API uses HTTP methods to perform operations. Some of the HTTP operations which doesn't modify the resource at the server is known as safe operations e.g. GET and HEAD. On the other hand, [PUT](http://javarevisited.blogspot.sg/2016/10/difference-between-put-and-post-in-restful-web-service.html), POST, and DELETE are unsafe because they modify the resource on the server.  
  
 **What are idempotent operations? Why is idempotency important?**([answer](http://javarevisited.blogspot.sg/2016/05/what-are-idempotent-and-safe-methods-of-HTTP-and-REST.html))  
There are some HTTP methods e.g. GET which produce same response no matter how many times you use them e.g. sending multiple GET request to the same URI will result in same response without any side-effect hence it is known as idempotent.  
  
On the other hand, the POST is not idempotent because if you send multiple POST request, it will result in multiple resource creation on the server, but again, PUT is idempotent if you are using it to update the resource.  
  
Even, multiple PUT request to update a resource on a server will give same end result. You can further take [HTTP Fundamentals](http://pluralsight.pxf.io/c/1193463/424552/7490?u=https%3A%2F%2Fwww.pluralsight.com%2Fcourses%2Fxhttp-fund) course by Pluralsight to learn more about idempotent methods of HTTP protocol and HTTP in general.

**Is REST scalable and/or interoperable?**(answer)  
Yes, REST is Scalable and interoperable. It doesn't mandate a specific choice of technology either at client or server end. You can use Java, C++, Python or JavaScript to create RESTful Web Services and Consume them at the client end. I suggest you read a good book on REST API e.g. [RESTful Web Services](http://javarevisited.blogspot.sg/2017/02/top-5-books-to-learn-rest-and-restful-web-services-in-java.html) to learn more about REST.  
  
  
**What are the advantages of the RestTemplate?**([answer](http://javarevisited.blogspot.sg/2017/02/how-to-consume-json-from-restful-web-services-Spring-RESTTemplate-Example.html))  
The RestTemplate class is an implementation of Template method pattern in Spring framework. Similar to other popular template classes e.g. JdbcTemplate or JmsTempalte, it also simplifies the interaction with RESTful Web Services on the client side. You can use it to consume a RESTful Web Servicer very easily as shown in this example.  
  
  
**Which HTTP methods does REST use?**([answer](http://javarevisited.blogspot.sg/2016/04/what-is-purpose-of-http-request-types-in-RESTful-web-service.html#axzz56WGunSwy))  
REST can use any HTTP methods but the most popular ones are GET for retrieving a resource, POST for creating a resource, PUt for updating resource and DELETE for removing a resource from the server.  
  
  
  
**What is an HttpMessageConverter in Spring REST?**(answer)  
An HttpMessageConverter is a [Strategy interface](http://www.java67.com/2014/12/strategy-pattern-in-java-with-example.html) that specifies a converter that can convert from and to HTTP requests and responses. Spring REST uses this interface to convert HTTP response to various formats e.g. JSON or XML.  
  
Each HttpMessageConverter implementation has one or several MIME Types associated with it. Spring uses the "Accept" header to determine the content type client is expecting.  
  
It will then try to find a registered HTTPMessageConverter that is capable of handling that specific content-type and use it to convert the response into that format before sending to the client.  
  
  
  
**How to create a custom implementation of HttpMessageConverter to support a new type of request/responses?**(answer)  
You just need to create an implementation of AbstractHttpMessageConverter and register it using the WebMvcConfigurerAdapter#extendMessageConverters() method with the classes which generate a new type of request/response.  
  
  
**Is REST normally stateless?**([answer](http://javarevisited.blogspot.sg/2015/08/difference-between-soap-and-restfull-webservice-java.html))  
Yes, REST API should be stateless because it is based on HTTP which is also stateless. A Request in REST API should contain all the details required it to process i.e. it should not rely on previous or next request or some data maintained at the server end e.g. Sessions. REST specification put a constraint to make it stateless and you should keep that in mind while designing your REST API.  
  
 **What does @RequestMapping annotation do?**([answer](http://javarevisited.blogspot.sg/2017/06/how-spring-mvc-framework-works-web-flow.html#axzz55vF5ugU8))  
The @RequestMapping annotation is used to map web requests to Spring Controller methods. You can map request based upon HTTP methods  e.g. GET and POST and various other parameters. For examples, if you are developing RESTful Web Service using Spring then you can use produces and consumes property along with media type annotation to indicate that this method is only used to produce or consumers JSON as shown below:

@RequestMapping (**method** **=** RequestMethod.POST, consumes**=**"application/json")

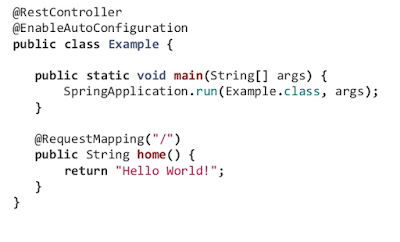
**public** Book save(@RequestBody Book aBook) {

**return** bookRepository.save(aBook);

}

You can similarly create other handler methods to produce JSON or XML. If you are not familiar with these annotations then I suggest you join [**Spring MVC For Beginners**](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-mvc-tutorial-for-beginners-step-by-step%2F) course on Udemy to learn from scratch.

**Is @Controller a stereotype? Is @RestController a stereotype?**([answer](http://javarevisited.blogspot.sg/2017/08/difference-between-restcontroller-and-controller-annotations-spring-mvc-rest.html))  
Yes, both @Controller and @RestController are stereotypes. The @Controller is actually a specialization of Spring's @Component stereotype annotation. This means that class annotated with @Controller will also be automatically be detected by Spring container as part of container's component scanning process.  
  
And, @RestController is a specialization of @Controller for RESTful web service. It not only combines @ResponseBody and @Controller annotation but also gives more meaning to your controller class to clearly indicate that it deals with RESTful requests.  
  
Spring Framework may also use this annotation to provide some more useful features related to REST API development in future.  
 **What is the difference between @Controller and @RestController?**([answer](http://javarevisited.blogspot.sg/2017/08/difference-between-restcontroller-and-controller-annotations-spring-mvc-rest.html))  
There are many differences between @Controller and @RestController as discussed in my earlier article (see the answer) but the most important one is that with @RestController you get the @ResponseBody annotation automatically, which means you don't need to separately annotate your handler methods with @ResponseBody annotation. This makes the development of RESTful web service easier using Spring. You can see here to learn

[](http://javarevisited.blogspot.sg/2016/12/top-5-spring-and-hibernate-training-courses-java-jee-programmers.html)

**When do you need @ResponseBody annotation in Spring MVC?**([answer](http://javarevisited.blogspot.sg/2018/01/7-reasons-for-using-spring-to-develop-RESTful-web-service.html#axzz55a8rTeu7))  
The @ResponseBody annotation can be put on a method to indicates that the return type should be written directly to the HTTP response body (and not placed in a Model, or interpreted as a view name).  
  
For example:  
  
@RequestMapping(path = "/hello", method = RequestMethod.PUT)  
@ResponseBody  
public String helloWorld() {  
   return "Hello World";  
}  
  
Alternatively, you can also use @RestController annotation instead of @Controller annotation. This will remove the need for using @ResponseBody because as discussed in the previous answer, it comes automatically with @RestController annotation.  
  
  
  
**What does @PathVariable do in Spring MVC? Why it's useful in REST with Spring?**([answer](http://javarevisited.blogspot.sg/2017/10/differences-between-requestparam-and-pathvariable-annotations-spring-mvc.html))  
It's one of the useful annotations from Spring MVC which allows you to read values from URI like query parameter. It's particularly useful in case of creating RESTful web service using Spring because in REST resource identifiers are part of URI.This questions is normally asked to experienced Spring MVC developers e.g. 4 to 6 years of experience.  
  
For example, in the URL http://myapp.com/books/101 if you want to extract 101 the id, then you can use @PathVariable annotation of Spring MVC.  If you are not familiar with Spring MVC annotations then [Spring MVC For Beginners: Build Java Web App in 25 Steps](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-mvc-tutorial-for-beginners-step-by-step%2F) is a good place to start with.

Overview

The @RequestMapping and @PathVariable annotations are widely used in Spring MVC so we can create controllers that accept dynamic values in the URI. This allows them to be used in the body of the method without the need for manual conversion or parsing.

In this article, we'll explain how these annotations work and how we can use them in a Spring web application.

How Request Mapping Works

Every HTTP request in a Spring MVC web app is routed through the DispatcherServlet or Front controller.  This servlet is responsible for routing incoming requests to handler methods of controllers.

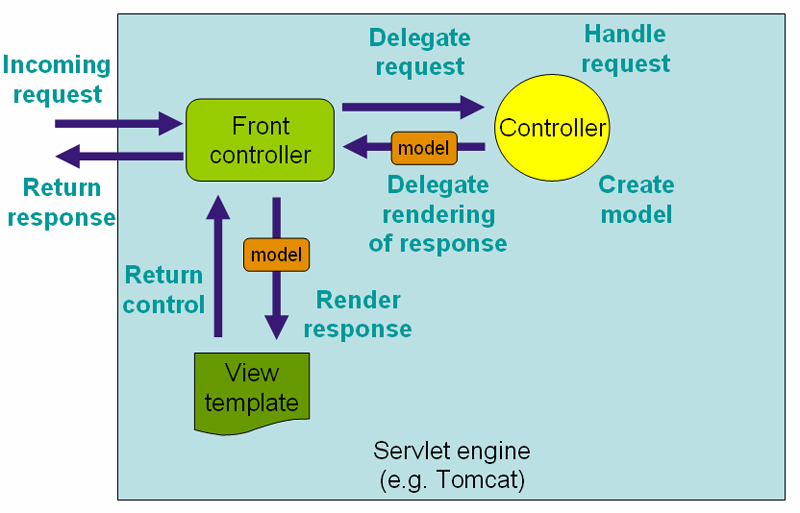


Diagram from Spring documentation at spring.io

Spring MVC Annotations

There are several annotations that allow us to configure our controllers to handle specific requests.

@RequestMapping

We specify the mappings between the requests and handler methods using the @RequestMapping annotation.

This annotation can be applied at the class or method level.  It maps a specific request path or pattern onto a controller.  If we annotate the class, we can use additional method-level annotations to make mappings more specific.

Code

@RequestMapping(value="/orders", method=RequestMethod.GET)

public String getOrders() {

return "All orders";

}

This method will be executed whenever the app receives a GET request at /orders.

URI Template Patterns

Template patterns are used to tell Spring which part of the URI we want to extract.

A template variable is enclosed with curly braces:

@RequestMapping(value="/orders/{id}")

There can be multiple template variables in the same URI:

@RequestMapping(value="/orders/{orderId}/items/{itemId}")

Shortcut Annotations

Spring 4.3 introduced some shorthand annotations for @RequestMapping that are easier to read:

* @GetMapping("/orders")
  + Equivalent to @RequestMapping(value="/orders", method=RequestMethod.GET)
* @GetMapping("/orders")
  + Equivalent to @RequestMapping(value="/orders", method=RequestMethod.GET)
* @PostMapping("/orders")
  + Equivalent to @RequestMapping(value="/orders", method=RequestMethod.POST)
* @PutMapping("/orders/{id}")
  + Equivalent to @RequestMapping(value="/orders{id}", method=RequestMethod.PUT)
* @DeleteMapping("/orders/{id}")
  + Equivalent to @RequestMapping(value="/orders{id}", method=RequestMethod.DELETE)
* @PatchMapping("/orders/{id}")
  + Equivalent to @RequestMapping(value="/orders{id}", method=RequestMethod.PATCH)

@PathVariable

This annotation is used on the method parameter we want to populate:

@RequestMapping(value = "/orders/{id}", method = RequestMethod.GET)

@ResponseBody

public String getOrder(@PathVariable final String id) {

return "Order ID: " + id;

}

Note that the template variable name (between the curly braces) and the parameter name should match.  If we wanted to use a different parameter name, we could specify the template variable name in the annotation:

@RequestMapping(value = "/orders/{id}", method = RequestMethod.GET)

@ResponseBody

public String getOrder(@PathVariable("id") final String orderId) {

return "Order ID: " + orderId;

}

@PathVariable can be used in any type of request method (GET, POST, DELETE, etc).

Example Controller Code

package com.codebyamir.demo.controller;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.ResponseBody;

@Controller

@RequestMapping(value = "/orders", method=RequestMethod.GET)

public class DemoController {

@RequestMapping(method = RequestMethod.GET)

@ResponseBody

public String getAllOrders() {

return "All orders";

}

@RequestMapping(value = "/{orderId}", method = RequestMethod.GET)

@ResponseBody

public String getOrder(@PathVariable final String orderId) {

return "Order ID: " + orderId;

}

@RequestMapping(value = "/{orderId}/items", method=RequestMethod.GET)

@ResponseBody

public String getItemsByOrder(@PathVariable final String orderId) {

return "Items for Order ID " + orderId;

}

@RequestMapping(value = "/{orderId}/items/{itemId}", method=RequestMethod.GET)

@ResponseBody

public String getItem(@PathVariable final String orderId, @PathVariable final String itemId) {

return "Order ID: " + orderId + ", Item ID: " + itemId;

}

}

This controller supports the following request URI's:

GET /orders  
GET /orders/100  
GET /orders/100/items  
GET /orders/100/items/1

Frequently Asked Questions (FAQ)

What’s the difference between @PathVariable, @PathParam, and @RequestParam?

Even though @PathVariable and @RequestParam are both used to extract values from the URL, their usage is largely determined by how a site is designed.

The @PathVariable annotation is used for data passed in the URI (e.g. RESTful web services) while @RequestParam is used to extract the data found in query parameters.

If we had chosen to use query parameters, our URL would be:

http://localhost:8080/orders?id=100

This would be implemented in a controller method like the following:

@GetMapping("/orders")

@ResponseBody

public String getOrder(@RequestParam(value = "id", required = true) String id) {

  return "Order ID: " + id;

}

These annotations can be mixed together inside the same controller.

@PathParam is a JAX-RS annotation that is equivalent to @PathVariable in Spring.

Missing URI template variable 'X' for method parameter of type Y

Spring will throw this exception if it cannot match up the template variable with the name of our path variable.

Let's take a look at an example causes this error:

@GetMapping("/orders/{id}")

public String getOrder(@PathVariable String orderId) {}

Under the hood, Spring looks at the name of our path variable orderId and tries to match it up with a template variable.  It cannot because our template variable is defined as id.

We can fix this by changing the path variable name to id.

Alternatively, @PathVariable accepts a value attribute:

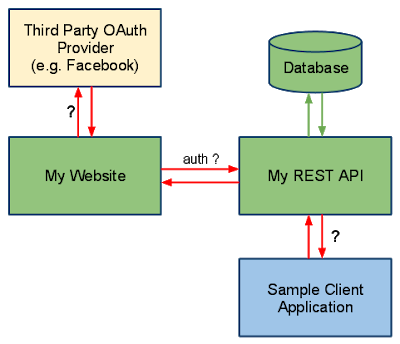
@GetMapping("/orders/{id}")

public String getOrder(@PathVariable("id") String orderId) {}

**What is the HTTP status return code for a successful DELETE statement?**([answer](http://www.java67.com/2015/09/top-10-restful-web-service-interview-questions-answers.html))  
There is no strict rule with respect to what status code your REST API should return after a successful DELETE i.e it can return 200 Ok or 204 No Content. In general, if the DELETE operation is successful and the response body is empty return 204. If the DELETE request is successful and the response body is NOT empty, return 200  
  
  
**What does CRUD mean?**(answer)  
CRUD is a short form of Create, Read, Update and Delete. In REST API, the POST is used to create a resource, GET is used to read a resource, [PUT](http://javarevisited.blogspot.sg/2016/10/difference-between-put-and-post-in-restful-web-service.html) is used to updated a resource and DELETE is used to remove a resource from the server. This one is another beginner level Spring MVC questions for 1 to 3 years experienced programmers  
  
  
**Where do you need @EnableWebMVC?**(answer)  
The @EnableWebMvc annotation is required to enable Spring MVC when Java configuration is used to configure Spring MVC instead of XML. It is equivalent to <mvc: annotation-driven>  in XML configuration.  
  
It enables support for @Controller-annotated classes that use @RequestMapping to map incoming requests to handler methods not already familiar with Spring's support for Java configuration, [Spring Master Class](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-tutorial-for-beginners%2F) on Udemy is a good place to start.

**When do you need @ResponseStatus annotation in Spring MVC?**([answer](http://javarevisited.blogspot.sg/2018/01/7-reasons-for-using-spring-to-develop-RESTful-web-service.html#axzz55a8rTeu7))

A good questions for 3 to 5 years experienced spring developers. The @ResponseStatus annotation is required during error handling in Spring MVC and REST. Normally when an error or exception is thrown at server side, web server return a blanket HTTP status code 500 - Internal server error.  
  
This may work for a human user but not for REST clients. You need to send them proper status code e.g. 404 if the resource is not found. That's where you can use @ResponseStatus annotation, which allows you to send custom HTTP status code along with proper error message in case of Exception.  
  
In order to use it, you can create custom exceptions and annotated them using @ResponseStatus annotation and proper HTTP status code and reason.  
  
When such exceptions are thrown from controller's handler methods and not handled anywhere else, then appropriate HTTP response with the proper HTTP status code, which you have set is sent to the client.  
  
For example, if you are writing a RESTful Web Service for a library which provides book information then you can use @ResponseStatus to create Exception which returns HTTP response code 404 when a book is not found instead of Internal Server Error (500), as shown below:  
  
 @ResponseStatus(value=HttpStatus.NOT\_FOUND, reason="No such Book")  // 404  
 public class BookNotFoundException extends RuntimeException {  
     // ...  
 }  
  
If this Exception is thrown from any handler method then HTTP error code 404 with reason "No such Book" will be returned to the client.  
  
  
**Is REST secure? What can you do to secure it?**([answer](http://www.java67.com/2017/04/3-great-books-to-learn-java-web-services-soap-and-restful.html))  
This question is mostly asked with experienced Java programmers e.g. 2 to 5 years experience with both REST and Spring. Security is a broad term, it could mean security of message which is provided by encryption or access restriction which is provided using authentication and authorization. REST is normally not secure but you can secure it by using Spring security.  
  
At the very least you can enable HTTP basic authentication by using HTTP in your Spring security configuration file. Similarly, you can expose your REST API using[HTTPS](http://javarevisited.blogspot.sg/2013/07/how-ssl-https-and-certificates-works-in-java-web-application.html) if the underlying server supports HTTPS.

[](http://javarevisited.blogspot.sg/2018/01/how-to-enable-http-basic-authentication-spring-security-java-xml-configuration.html)

**Does REST work with transport layer security (TLS)?**([answer](http://javarevisited.blogspot.sg/2012/01/rest-web-services-framework-interview.html))  
TLS or Transport Layer Security is used for secure communication between client and server. It is the successor of SSL (Secure Socket Layer). Since HTTPS can work with both SSL and TLS, REST can also work with TLS.  
  
Actually, REST says anything about Security, it's up to the server which implements that. Same RESTful Web Service can be accessed using HTTP and HTTPS if the server supports [SSL](http://javarevisited.blogspot.sg/2013/07/how-to-configure-https-ssl-in-tomcat-6-7-web-server-java.html#axzz56WXxxAC0).  
  
If you are using Tomcat, you can see here to learn more about how to enable SSL in Tomcat.  
  
  
**Do you need Spring MVC in your classpath for developing RESTful Web Service?**([answer](http://javarevisited.blogspot.sg/2017/01/where-and-how-to-download-spring-JAR-Files-Spring4-without-Maven-Gradle.html#axzz4pp42TeHu))  
This question is often asked to Java programmers with 1 to 2 years of experience in Spring. Short answer is Yes, you need Spring MVC in your Java application's classpath to develop RESTful web services using Spring framework. It's actually Spring MVC which provides all useful annotations e.g. @RestController, @ResponseCode, @ResponseBody, @RequestBody, and @PathVariable, hence you must spring-mvc.jar or appropriate Maven entry in your pom.xml  
  
  
That's all about some **frequently asked Spring REST Interview questions** for beginners and experienced Java JEE developers. These questions are also very useful to brush up your knowledge about Spring REST if you are going to take Spring Certification. If you need more questions from Spring certification perspective, you will find a lot of question on this topic on David Mayer's [Core Spring Simulator](https://certification-questions.com/spring-free-mock-exams/spring-core-v4.2-practice-test.html?affiliateCode=fcff36fd-557a-4713-abf6-973e9924770f&utm_source=Javin&utm_medium=affiliate&utm_campaign=affiliate), one of the best simulator to pass Spring certification at the moment.

1. [What is a Web Service?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#what-is-web-service)
2. [What are the advantages of Web Services?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#advantages-of-web-services)
3. [What are different types of Web Services?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#types-of-web-services)
4. [What is SOAP?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#soap-definition)
5. [What are advantages of SOAP Web Services?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#soap-advantages)
6. [What are disadvantages of SOAP Web Services?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#soap-disadvantages)
7. [What is WSDL?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#wsdl-definition)
8. [What are different components of WSDL?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#wsdl-components)
9. [What is UDDI?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#uddi-definition)
10. [What is difference between Top Down and Bottom Up approach in SOAP Web Services?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#soap-ws-approach)
11. [What is REST Web Services?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#rest-web-services)
12. [What are advantages of REST web services?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#rest-advantages)
13. [What are disadvantages of REST web services?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#rest-disadvantages)
14. [What is a Resource in Restful web services?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#restful-resource)
15. [What are different HTTP Methods supported in Restful Web Services?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#http-methods)
16. [Compare SOAP and REST web services?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#soap-vs-rest)
17. [What are different ways to test web services?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#web-services-testing)
18. [Can we maintain user session in web services?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#web-services-seesion)
19. [What is difference between SOA and Web Services?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#soa-vs-web-services)
20. [What is the use of Accept and Content-Type Headers in HTTP Request?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#http-request-headers)
21. [How would you choose between SOAP and REST web services?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#soap-rest-choice)
22. [What is JAX-WS API?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#jax-ws-api)
23. [Name some frameworks in Java to implement SOAP web services?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#java-soap-frameworks)
24. [Name important annotations used in JAX-WS API?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#jax-ws-annotations)
25. [What is use of javax.xml.ws.Endpoint class?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#jax-ws-endpoint)
26. [What is the difference between RPC Style and Document Style SOAP web Services?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#soap-rpc-vs-document)
27. [How to get WSDL file of a SOAP web service?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#soap-wsdl-url)
28. [What is sun-jaxws.xml file?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#soap-config-file)
29. [What is JAX-RS API?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#jax-rs-api)
30. [Name some implementations of JAX-RS API?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#jax-rs-implementations)
31. [What is wsimport utility?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#wsimport-utility)
32. [Name important annotations used in JAX-RS API?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#jax-rs-annotations)
33. [What is the use of @XmlRootElement annotation?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#xmlrootelement-jaxb)
34. [How to set different status code in HTTP response?](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful#http-response-code)
35. **What is a Web Service?**

Web Services work on client-server model where client applications can access web services over the network. Web services provide endpoint URLs and expose methods that can be accessed over network through client programs written in java, shell script or any other different technologies.  
Web services are stateless and doesn’t maintain user session like web applications.

1. **What are the advantages of Web Services?**

Some of the advantages of web services are:

* + Interoperability: Web services are accessible over network and runs on HTTP/SOAP protocol and uses XML/JSON to transport data, hence it can be developed in any programming language. Web service can be written in java programming and client can be PHP and vice versa.
  + Reusability: One web service can be used by many client applications at the same time.
  + Loose Coupling: Web services client code is totally independent with server code, so we have achieved loose coupling in our application.
  + Easy to deploy and integrate, just like web applications.
  + Multiple service versions can be running at same time.



1. **What are different types of Web Services?**

There are two types of web services:

* + SOAP Web Services: Runs on SOAP protocol and uses XML technology for sending data.
  + Restful Web Services: It’s an architectural style and runs on HTTP/HTTPS protocol almost all the time. REST is a stateless client-server architecture where web services are resources and can be identified by their URIs. Client applications can use HTTP GET/POST methods to invoke Restful web services.

1. **What is SOAP?**

SOAP stands for Simple Object Access Protocol. SOAP is an XML based industry standard protocol for designing and developing web services. Since it’s XML based, it’s platform and language independent. So our server can be based on JAVA and client can be on .NET, PHP etc. and vice versa.

1. **What are advantages of SOAP Web Services?**

SOAP web services have all the advantages that web services has, some of the additional advantages are:

* + WSDL document provides contract and technical details of the web services for client applications without exposing the underlying implementation technologies.
  + SOAP uses XML data for payload as well as contract, so it can be easily read by any technology.
  + SOAP protocol is universally accepted, so it’s an industry standard approach with many easily available open source implementations.

1. **What are disadvantages of SOAP Web Services?**

Some of the disadvantages of SOAP protocol are:

* + Only XML can be used, JSON and other lightweight formats are not supported.
  + SOAP is based on the contract, so there is a tight coupling between client and server applications.
  + SOAP is slow because payload is large for a simple string message, since it uses XML format.
  + Anytime there is change in the server side contract, client stub classes need to be generated again.
  + Can’t be tested easily in browser

1. **What is WSDL?**

WSDL stands for Web Service Description Language. WSDL is an XML based document that provides technical details about the web service. Some of the useful information in WSDL document are: method name, port types, service end point, binding, method parameters etc.

1. **What are different components of WSDL?**

Some of the different tags in WSDL xml are:

* + xsd:import namespace and schemaLocation: provides WSDL URL and unique namespace for web service.
  + message: for method arguments
  + part: for method argument name and type
  + portType: service name, there can be multiple services in a wsdl document.
  + operation: contains method name
  + soap:address for endpoint URL.

1. **What is UDDI?**

UDDI is acronym for Universal Description, Discovery and Integration. UDDI is a directory of web services where client applications can lookup for web services. Web Services can register to the UDDI server and make them available to client applications.

1. **What is difference between Top Down and Bottom Up approach in SOAP Web Services?**

In Top Down approach first WSDL document is created to establish the contract between web service and client and then code is written, it’s also termed as contract first approach. This is hard to implement because classes need to be written to confirm the contract established in WSDL. Benefit of this approach is that both client and server code can be written in parallel.

In Bottom Up approach, first web service code is written and then WSDL is generated. It’s also termed as contract last approach. This approach is easy to implement because WSDL is generated based on code. In this approach client code have to wait for WSDL from server side to start their work.

1. **What is REST Web Services?**

REST is the acronym for REpresentational State Transfer. REST is an architectural style for developing applications that can be accessed over the network. REST architectural style was brought in light by Roy Fielding in his doctoral thesis in 2000.

REST is a stateless client-server architecture where web services are resources and can be identified by their URIs. Client applications can use HTTP GET/POST methods to invoke Restful web services. REST doesn’t specify any specific protocol to use, but in almost all cases it’s used over HTTP/HTTPS. When compared to SOAP web services, these are lightweight and doesn’t follow any standard. We can use XML, JSON, text or any other type of data for request and response.

1. **What are advantages of REST web services?**

Some of the advantages of REST web services are:

* + Learning curve is easy since it works on HTTP protocol
  + Supports multiple technologies for data transfer such as text, xml, json, image etc.
  + No contract defined between server and client, so loosely coupled implementation.
  + REST is a lightweight protocol
  + REST methods can be tested easily over browser.

1. **What are disadvantages of REST web services?**

Some of the disadvantages of REST are:

* + Since there is no contract defined between service and client, it has to be communicated through other means such as documentation or emails.
  + Since it works on HTTP, there can’t be asynchronous calls.
  + Sessions can’t be maintained.

1. **What is a Resource in Restful web services?**

Resource is the fundamental concept of Restful architecture. A resource is an object with a type, relationship with other resources and methods that operate on it. Resources are identified with their URI, HTTP methods they support and request/response data type and format of data.

1. **What are different HTTP Methods supported in Restful Web Services?**

Restful web services supported HTTP methods are – GET, POST, PUT, DELETE and HEAD.

1. **Compare SOAP and REST web services?**

|  |  |
| --- | --- |
| SOAP | REST |
|  |  |
| SOAP is a standard protocol for creating web services. | REST is an architectural style to create web services. |
| SOAP is acronym for Simple Object Access Protocol. | REST is acronym for REpresentational State Transfer. |
| SOAP uses WSDL to expose supported methods and technical details. | REST exposes methods through URIs, there are no technical details. |
| SOAP web services and client programs are bind with WSDL contract | REST doesn’t have any contract defined between server and client |
| SOAP web services and client are tightly coupled with contract. | REST web services are loosely coupled. |
| SOAP learning curve is hard, requires us to learn about WSDL generation, client stubs creation etc. | REST learning curve is simple, POJO classes can be generated easily and works on simple HTTP methods. |
| SOAP supports XML data format only | REST supports any data type such as XML, JSON, image etc. |
| SOAP web services are hard to maintain, any change in WSDL contract requires us to create client stubs again and then make changes to client code. | REST web services are easy to maintain when compared to SOAP, a new method can be added without any change at client side for existing resources. |
| SOAP web services can be tested through programs or software such as Soap UI. | REST can be easily tested through CURL command, Browsers and extensions such as Chrome Postman. |

## SOAP vs. REST

Let' have a quick overview of SOAP and REST before we do a deep dive into the key differences between them.

**SOAP** – SOAP is a protocol which was designed before REST and came into the picture. The main idea behind designing SOAP was to ensure that programs built on different platforms and programming languages could exchange data in an easy manner.

**REST** – This was designed specifically for working with components such as media components, files, or even objects on a particular hardware device. Any web service that is defined on the principles of REST can be called a RestFul web service. A Restful service would use the normal HTTP verbs of GET, POST, PUT and DELETE for working with the required components.

Below are the main differences between SOAP and REST

|  |  |
| --- | --- |
| **SOAP** | **REST** |
| * SOAP stands for Simple Object Access Protocol | * REST stands for Representational State Transfer |
| * SOAP is a protocol. SOAP was designed with a specification. It includes a WSDL file which has the required information on what the web service does in addition to the location of the web service. | * REST is an Architectural style in which a web service can only be treated as a RESTful service if it follows the constraints of being   1. Client Server   2. Stateless   3. Cacheable   4. Layered System   5. Uniform Interface |
| * SOAP cannot make use of REST since SOAP is a protocol and REST is an architectural pattern. | * REST can make use of SOAP as the underlying protocol for web services, because in the end it is just an architectural pattern. |
| * SOAP uses service interfaces to expose its functionality to client applications. In SOAP, the WSDL file provides the client with the necessary information which can be used to understand what services the web service can offer. | * REST use Uniform Service locators to access to the components on the hardware device. For example, if there is an object which represents the data of an employee hosted on a URL as http://demo.guru99 , the below are some of URI that can exist to access them   http://demo.guru99.com/Employee  http://demo.guru99.com/Employee/1 |
| * SOAP requires more bandwidth for its usage. Since SOAP Messages contain a lot of information inside of it, the amount of data transfer using SOAP is generally a lot.   <?xml version="1.0"?>  <SOAP-ENV:Envelope  xmlns:SOAP-ENV  ="http://www.w3.org/2001/12/soap-envelope"  SOAP-ENV:encodingStyle  =" http://www.w3.org/2001/12/soap-encoding">  <soap:Body>  <Demo.guru99WebService  xmlns="http://tempuri.org/">  <EmployeeID>int</EmployeeID>  </Demo.guru99WebService>  </soap:Body>  </SOAP-ENV:Envelope> | * REST does not need much bandwidth when requests are sent to the server. REST messages mostly just consist of JSON messages. Below is an example of a JSON message passed to a web server. You can see that the size of the message is comparatively smaller to SOAP.   {"city":"Mumbai","state":"Maharastra"} |
| * SOAP can only work with XML format. As seen from SOAP messages, all data passed is in XML format. | * REST permits different data format such as Plain text, HTML, XML, JSON, etc. But the most preferred format for transferring data is JSON. |

## When to use REST and when to use SOAP

One of the most highly debatable topics is when REST should be used or when to use SOAP while designing web services.

Below are some of the key factors that determine when each technology should be used for web services **REST services should be used in the following instances**

* **Limited resources and bandwidth** – Since SOAP messages are heavier in content and consume a far greater bandwidth, REST should be used in instances where network bandwidth is a constraint.
* **Statelessness** – If there is no need to maintain a state of information from one request to another then REST should be used. If you need a proper information flow wherein some information from one request needs to flow into another then SOAP is more suited for that purpose. We can take the example of any online purchasing site. These sites normally need the user first to add items which need to be purchased to a cart. All of the cart items are then transferred to the payment page in order to complete the purchase. This is an example of an application which needs the state feature. The state of the cart items needs to be transferred to the payment page for further processing.
* **Caching**– If there is a need to cache a lot of requests then REST is the perfect solution. At times, clients could request for the same resource multiple times. This can increase the number of requests which are sent to the server. By implementing a cache, the most frequent queries results can be stored in an intermediate location. So whenever the client requests for a resource, it will first check the cache. If the resources exist then, it will not proceed to the server. So caching can help in minimizing the amount of trips which are made to the web server.
* **Ease of coding**– Coding REST Services and subsequent implementation is far easier than SOAP. So if a quick win solution is required for web services, then REST is the way to go.

SOAP should be used in the following instances

1. **Asynchronous processing and subsequent invocation** – if there is a requirement that the client needs a guaranteed level of reliability and security then the new SOAP standard of SOAP 1.2 provides a lot of additional features, especially when it comes to security.
2. **A Formal means of communication** – if both the client and server have an agreement on the exchange format then SOAP 1.2 gives the rigid specifications for this type of interaction. An example is an online purchasing site in which users add items to a cart before the payment is made. Let's assume we have a web service that does the final payment. There can be a firm agreement that the web service will only accept the cart item name, unit price, and quantity. If such a scenario exists then, it's always better to use the SOAP protocol.
3. **Stateful operations –**ifthe application has a requirement that state needs to be maintained from one request to another, then the SOAP 1.2 standard provides the WS\* structure to support such requirements.

## SOAP vs. REST API challenges

API is known as the **Application Programming Interface** and is offered by both the client and the server. In the client world, this is offered by the browser whereas in the server world it's what is provided by the web service which can either be SOAP or REST.

**Challenges with the SOAP API**

1. WSDL file - One of the key challenges of the SOAP API is the WSDL document itself. The WSDL document is what tells the client of all the operations that can be performed by the web service. The WSDL document will contain all information such as the data types being used in the SOAP messages and what all operations are available via the web service. The below code snippet is just part of a sample WSDL file.

<?xml version="1.0"?>

<definitions name="Tutorial"

targetNamespace=http://demo.guru99.com/Tutorial.wsdl

xmlns:tns=http://demo.guru99.com/Tutorial.wsdl

xmlns:xsd1=http://demo.guru99.com/Tutorial.xsd

xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap/

xmlns="http://schemas.xmlsoap.org/wsdl/">

<types>

<schema targetNamespace=http://Demo.guru99.com/Tutorial.xsd

xmlns="http://www.w3.org/2000/10/XMLSchema">

<element name="TutorialNameRequest">

<complexType>

<all>

<element name="TutorialName" type="string"/>

</all>

</complexType>

</element>

<element name="TutorialIDRequest">

<complexType>

<all>

<element name="TutorialID" type="number"/>

</all>

</complexType>

</element>

</schema>

</types>

As per the above WSDL file, we have an element called "TutorialName" which is of the type String which is part of the element TutorialNameRequest.

Now, suppose if the WSDL file were to change as per the business requirements and the TutorialName has to become TutorialDescription. This would mean that all the clients who are currently connecting to this web service would then need to make this corresponding change in their code to accommodate the change in the WSDL file.

This shows the biggest challenge of the WSDL file which is the tight contract between the client and the server and that one change could cause a large impact, on the whole, client applications.

1. Document size – The other key challenge is the size of the SOAP messages which get transferred from the client to the server. Because of the large messages, using SOAP in places where bandwidth is a constraint can be a big issue.

**Challenges with the REST API**

1. **Lack of Security**– REST does not impose any sort of security like SOAP. This is why REST is very appropriate for public available URL's, but when it comes down to confidential data being passed between the client and the server, REST is the worst mechanism to be used for web services.
2. **Lack of state**– Most web applications require a stateful mechanism. For example, if you had a purchasing site which had the mechanism of having a shopping cart, it is required to know the number of items in the shopping cart before the actual purchase is made. Unfortunately, the burden of maintaining this state lies with the client, which just makes the client application heavier and difficult to maintain.

### What are different ways to test web services?

SOAP web services can be tested programmatically by generating client stubs from WSDL or through software such as Soap UI.

REST web services can be tested easily with program, curl commands and through browser extensions. Resources supporting GET method can be tested with browser itself, without any program.

### Can we maintain user session in web services?

Web services are stateless so we can’t maintain user sessions in web services.

### What is difference between SOA and Web Services?

Service Oriented Architecture (SOA) is an architectural pattern where applications are designed in terms of services that can be accessed through communication protocol over network. SOA is a [design pattern](https://www.journaldev.com/1827/java-design-patterns-example-tutorial) and doesn’t go into implementation.

Web Services can be thought of as Services in SOAP architecture and providing means to implement SOA pattern.

### What is the use of Accept and Content-Type Headers in HTTP Request?

These are important headers in Restful web services. Accept headers tells web service what kind of response client is accepting, so if a web service is capable of sending response in XML and JSON format and client sends Accept header as “application/xml” then XML response will be sent. For Accept header “application/json”, server will send the JSON response.

Content-Type header is used to tell server what is the format of data being sent in the request. If Content-Type header is “application/xml” then server will try to parse it as XML data. This header is useful in HTTP Post and Put requests.

### How would you choose between SOAP and REST web services?

Web Services work on client-server model and when it comes to choose between SOAP and REST, it all depends on project requirements. Let’s look at some of the conditions affecting our choice:

* + Do you know your web service clients beforehand? If Yes, then you can define a contract before implementation and SOAP seems better choice. But if you don’t then REST seems better choice because you can provide sample request/response and test cases easily for client applications to use later on.
  + How much time you have? For quick implementation REST is the best choice. You can create web service easily, test it through browser/curl and get ready for your clients.
  + What kind of data format are supported? If only XML then you can go with SOAP but if you think about supporting JSON also in future then go with REST.

### What is JAX-WS API?

JAX-WS stands for Java API for XML Web Services. JAX-WS is XML based Java API to build web services server and client application. It’s part of standard Java API, so we don’t need to include anything else which working with it. Refer to [JAX-WS Tutorial](https://www.journaldev.com/9123/jax-ws-tutorial) for a complete example.

### Name some frameworks in Java to implement SOAP web services?

We can create SOAP web services using JAX-WS API, however some of the other frameworks that can be used are Apache Axis and Apache CXF. Note that they are not implementations of JAX-WS API, they are totally different framework that work on Servlet model to expose your business logic classes as SOAP web services. Read more at [Java SOAP Web Service Eclipse](https://www.journaldev.com/9131/soap-webservices-in-java-example-eclipse) example.

### Name important annotations used in JAX-WS API?

Some of the important annotations used in JAX-WS API are:

* + @WebService
  + @SOAPBinding
  + @WebMethod

### What is use of javax.xml.ws.Endpoint class?

Endpoint class provides useful methods to create endpoint and publish existing implementation as web service. This comes handy in testing web services before making further changes to deploy it on actual server.

# HTTP Methods

RESTful APIs enable you to develop any kind of web application having all possible CRUD (create, retrieve, update, delete) operations. REST guidelines suggest using a specific HTTP method on a specific type of call made to the server (though technically it is possible to violate this guideline, yet it is highly discouraged).

Use below-given information to find suitable HTTP method for the action performed by API.

Table of Contents

[HTTP GET](https://restfulapi.net/http-methods/#get)

[HTTP POST](https://restfulapi.net/http-methods/#post)

[HTTP PUT](https://restfulapi.net/http-methods/#put)

[HTTP DELETE](https://restfulapi.net/http-methods/#delete)

[HTTP PATCH](https://restfulapi.net/http-methods/#patch)

[Summary](https://restfulapi.net/http-methods/#summary)

[Glossary](https://restfulapi.net/http-methods/#glossary)

## HTTP GET

Use GET requests **to retrieve resource representation/information only** – and not to modify it in any way. As GET requests do not change the state of the resource, these are said to be **safe methods**. Additionally, GET APIs should be **idempotent**, which means that making multiple identical requests must produce the same result every time until another API (POST or PUT) has changed the state of the resource on the server.

If the Request-URI refers to a data-producing process, it is the produced data which shall be returned as the entity in the response and not the source text of the process, unless that text happens to be the output of the process.

For any given HTTP GET API, if the resource is found on the server then it must return HTTP response code 200 (OK) – along with response body which is usually either XML or JSON content (due to their platform independent nature).

In case resource is NOT found on server then it must return HTTP response code 404 (NOT FOUND). Similarly, if it is determined that GET request itself is not correctly formed then server will return HTTP response code 400 (BAD REQUEST).

#### Example request URIs

* HTTP GET http://www.appdomain.com/users
* HTTP GET http://www.appdomain.com/users?size=20&page=5
* HTTP GET http://www.appdomain.com/users/123
* HTTP GET http://www.appdomain.com/users/123/address

## HTTP POST

Use POST APIs **to create new subordinate resources**, e.g. a file is subordinate to a directory containing it or a row is subordinate to a database table. Talking strictly in terms of REST, POST methods are used to create a new resource into the collection of resources.

Ideally, if a resource has been created on the origin server, the response SHOULD be HTTP response code 201 (Created) and contain an entity which describes the status of the request and refers to the new resource, and a [Location](https://en.wikipedia.org/wiki/HTTP_location) header.

Many times, the action performed by the POST method might not result in a resource that can be identified by a URI. In this case, either HTTP response code 200 (OK) or 204 (No Content) is the appropriate response status.

Responses to this method are **not cacheable**, unless the response includes appropriate [Cache-Control](https://en.wikipedia.org/wiki/Web_cache#Cache_control)or [Expires](https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html) header fields.

Please note that POST is **neither safe nor idempotent** and invoking two identical POST requests will result in two different resources containing the same information (except resource ids).

#### Example request URIs

* HTTP POST http://www.appdomain.com/users
* HTTP POST http://www.appdomain.com/users/123/accounts

## HTTP PUT

Use PUT APIs primarily **to update existing resource** (if the resource does not exist then API may decide to create a new resource or not). If a new resource has been created by the PUT API, the origin server MUST inform the user agent via the HTTP response code 201 (Created) response and if an existing resource is modified, either the 200 (OK) or 204 (No Content) response codes SHOULD be sent to indicate successful completion of the request.

If the request passes through a cache and the Request-URI identifies one or more currently cached entities, those entries SHOULD be treated as stale. Responses to this method are **not cacheable**.

*The difference between the POST and PUT APIs can be observed in request URIs. POST requests are made on resource collections whereas PUT requests are made on an individual resource.*

#### Example request URIs

* HTTP PUT http://www.appdomain.com/users/123
* HTTP PUT http://www.appdomain.com/users/123/accounts/456

## HTTP DELETE

As the name applies, DELETE APIs are used **to delete resources** (identified by the Request-URI).

A successful response of DELETE requests SHOULD be HTTP response code 200 (OK) if the response includes an entity describing the status, 202 (Accepted) if the action has been queued, or 204 (No Content) if the action has been performed but the response does not include an entity.

DELETE operations are **idempotent**. If you DELETE a resource, it’s removed from the collection of resource. Repeatedly calling DELETE API on that resource will not change the outcome – however calling DELETE on a resource a second time will return a 404 (NOT FOUND) since it was already removed. Some may argue that it makes DELETE method non-idempotent. It’s a matter of discussion and personal opinion.

If the request passes through a cache and the Request-URI identifies one or more currently cached entities, those entries SHOULD be treated as stale. Responses to this method are **not cacheable**.

#### Example request URIs

* HTTP DELETE http://www.appdomain.com/users/123
* HTTP DELETE http://www.appdomain.com/users/123/accounts/456

## HTTP PATCH

HTTP PATCH requests are **to make partial update on a resource**. If you see PUT requests also modify a resource entity so to make more clear – PATCH method is the correct choice for partially updating an existing resource and PUT should only be used if you’re replacing a resource in its entirety.

Please note that there are some challenges if you decide to use PATCH APIs in your application:

* Support for PATCH in browsers, servers, and web application frameworks is not universal. IE8, PHP, Tomcat, Django, and lots of other software has missing or broken support for it.
* Request payload of PATCH request is not straightforward as it is for PUT request. e.g.

HTTP GET /users/1

produces below response:

{id: 1, username: 'admin', email: 'email@example.org'}

A sample patch request to update the email will be like this:

HTTP PATCH /users/1

[  
{ “op”: “replace”, “path”: “/email”, “value”: “new.email@example.org” }  
]

There may be following possible operations are per HTTP specification.

[  
{ "op": "test", "path": "/a/b/c", "value": "foo" },  
{ "op": "remove", "path": "/a/b/c" },  
{ "op": "add", "path": "/a/b/c", "value": [ "foo", "bar" ] },  
{ "op": "replace", "path": "/a/b/c", "value": 42 },  
{ "op": "move", "from": "/a/b/c", "path": "/a/b/d" },  
{ "op": "copy", "from": "/a/b/d", "path": "/a/b/e" }  
]

PATCH method is not a replacement for the POST or PUT methods. It applies a delta (diff) rather than replacing the entire resource.

## Summary of HTTP Methods for RESTful APIs

Below table summarises the use of HTTP methods discussed above.

| **HTTP METHOD** | **CRUD** | **ENTIRE COLLECTION (E.G. /USERS)** | **SPECIFIC ITEM (E.G. /USERS/123)** |
| --- | --- | --- | --- |
| POST | Create | 201 (Created), ‘Location’ header with link to /users/{id} containing new ID. | Avoid using POST on single resource |
| GET | Read | 200 (OK), list of users. Use pagination, sorting and filtering to navigate big lists. | 200 (OK), single user. 404 (Not Found), if ID not found or invalid. |
| PUT | Update/Replace | 404 (Not Found), unless you want to update every resource in the entire collection of resource. | 200 (OK) or 204 (No Content). Use 404 (Not Found), if ID not found or invalid. |
| PATCH | Partial Update/Modify | 404 (Not Found), unless you want to modify the collection itself. | 200 (OK) or 204 (No Content). Use 404 (Not Found), if ID not found or invalid. |
| DELETE | Delete | 404 (Not Found), unless you want to delete the whole collection — use with caution. | 200 (OK). 404 (Not Found), if ID not found or invalid. |

# HTTP Status Codes

REST APIs use the **Status-Line** part of an HTTP response message to inform clients of their request’s overarching result. [RFC 2616](https://www.ietf.org/rfc/rfc2616.txt) defines the [Status-Line syntax](https://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6.1) as shown below:

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

HTTP defines forty standard status codes that can be used to convey the results of a client’s request. The status codes are divided into the five categories presented below.

|  |  |
| --- | --- |
| **CATEGORY** | **DESCRIPTION** |
| **1xx: Informational** | Communicates transfer protocol-level information. |
| **2xx: Success** | Indicates that the client’s request was accepted successfully. |
| **3xx: Redirection** | Indicates that the client must take some additional action in order to complete their request. |
| **4xx: Client Error** | This category of error status codes points the finger at clients. |
| **5xx: Server Error** | The server takes responsibility for these error status codes. |

Now look at subset of codes that specially apply to the design of a REST APIs – in some more detail.

#### [200 (OK)](https://restfulapi.net/http-status-200-ok/" \t "_blank)

It indicates that the REST API successfully carried out whatever action the client requested, and that no more specific code in the 2xx series is appropriate.

Unlike the 204 status code, a 200 response should include a response body.The information returned with the response is dependent on the method used in the request, for example:

* GET an entity corresponding to the requested resource is sent in the response;
* HEAD the entity-header fields corresponding to the requested resource are sent in the response without any message-body;
* POST an entity describing or containing the result of the action;
* TRACE an entity containing the request message as received by the end server.

#### [201 (Created)](https://restfulapi.net/http-status-201-created/" \t "_blank)

A REST API responds with the 201 status code whenever a resource is created inside a collection. There may also be times when a new resource is created as a result of some controller action, in which case 201 would also be an appropriate response.

The newly created resource can be referenced by the URI(s) returned in the entity of the response, with the most specific URI for the resource given by a Location header field.

The origin server MUST create the resource before returning the 201 status code. If the action cannot be carried out immediately, the server SHOULD respond with 202 (Accepted) response instead.

#### [202 (Accepted)](https://restfulapi.net/http-status-202-accepted/" \t "_blank)

A 202 response is typically used for actions that take a long while to process. It indicates that the request has been accepted for processing, but the processing has not been completed. The request might or might not be eventually acted upon, or even maybe disallowed when processing occurs.

Its purpose is to allow a server to accept a request for some other process (perhaps a batch-oriented process that is only run once per day) without requiring that the user agent’s connection to the server persist until the process is completed.

The entity returned with this response SHOULD include an indication of the request’s current status and either a pointer to a status monitor (job queue location) or some estimate of when the user can expect the request to be fulfilled.

#### [204 (No Content)](https://restfulapi.net/http-status-204-no-content/" \t "_blank)

The 204 status code is usually sent out in response to a PUT, POST, or DELETE request when the REST API declines to send back any status message or representation in the response message’s body.

An API may also send 204 in conjunction with a GET request to indicate that the requested resource exists, but has no state representation to include in the body.

If the client is a user agent, it SHOULD NOT change its document view from that which caused the request to be sent. This response is primarily intended to allow input for actions to take place without causing a change to the user agent’s active document view, although any new or updated metainformation SHOULD be applied to the document currently in the user agent’s active view.

The 204 response MUST NOT include a message-body and thus is always terminated by the first empty line after the header fields.

## Most Common HTTP Errors Explained & How to Fix Them

We’ve all seen them; those annoying messages that crop up on screen when we try and bring up a website. But do you know what they mean? We researched the many HTTP error messages in use, and the following is an explanation of each one in simple terms, together with advice on what to do when you are party to that message.

## 300 Series – Redirection Messages

### 300 HTTP Error (Multiple Choices)

This error message indicates that the particular site or other resource you are trying to access is no longer in that location. You will see the code plus a choice of new locations. The message could be caused by an incomplete URL or one that includes other locations within it; perhaps a list of ‘inner’ pages that may need to be individually selected.

**What to do:** in the first instance, enter the URL you are using in the browser. You may see a web page asking you for another action; if this is case, you are using a URL that the browser considers to be short on specific detail. Select one of the choices offered to proceed, or send the website a message if you are concerned.

### 301 HTTP Error (Moved Permanently)

This indicates a permanent relocation of the URL. The site or resource you are trying to access is no longer at that address, and the message should include the new location. Users should use the new address given in the message.

**What to do:** a 301 message should include the new URL; if so then the browser ought to retry accessing the new URL. Users should, in fact, not actually see a 301 message if it operates correctly, as reconnection should be automatic. There are occasions when corruption can lead to incorrect response from a 301 message. If this is so, contact the webmaster and inform them of the problem for ease of use in the future.

### 302 HTTP Error (Moved Temporarily)

The most commonly experienced redirection message, 302 means that the server believes the URL to have been redirected, temporarily, to an alternative address. The location of the new URL should be included in the message, and redirection should be automatic, as above.

**What to do:** as the required site or resource is located for the time being at an alternative address, the user should still use the original address and the request will be redirected. Alternatively, the server should show the new URL at which to access the information required.

### 303 HTTP Error (See Other)

This code is telling the user that the response can be found at a specific URL, as will be displayed. That URL should not be cached as the original URL has not been moved. The new URL is not, it should be stressed, a replacement for the one used; it simply is using a new URL for reasons the user may not be party to. This message may be seen quite frequently, as it is a much used method of redirection.

**What to do:** first and foremost, never cache the new URL, simply use it – if not automatically redirected – as the destination. if not using a HEAD request method, a hypertext note should be included in the response, or contact with the website at the relevant email address should be made.

### 304 HTTP Error (Not Modified)

Not strictly an error message, 304 merely tells the user that no changes have been made since the URL was accessed previously. Some web browsers do not allow for 304 messages, hence seeing one means that you browser does so. It is helpful in understanding whether information at the URL is up to date or has been modified.

**What to do:** a 304 message will, in most cases, send the user to the website requested as it is for information only, as it understands that the site has not been changed. Any problems, contact the website at its relevant email address.

### 305 HTTP Error (Use Proxy)

The 305 code is informing you that the site or resource you are requesting must be accessed via a proxy, and this will be given in the message. Basically, your web server believes that the URL you want is via proxy, possibly down to reasons of security. This can be a problematic code and is often dealt with incorrectly.

**What to do:** in the location field you will find the address of the proxy URL; simply use this to access the resource and bring up the information required.

### 306 HTTP Error

This 300 code is now defunct.

### 307 HTTP Error (Temporary Redirect)

Quite simply, a 307 error informs the user that the URL requested is temporarily available at an alternative URL, which will be outlined in the message itself. Users should redirect as instructed, but use the original URL in future.

**What to do:** use the temporary URL to access the required information; the message may include a hyperlink to the new URL for the use of the user.

## 400 Series – Client Error

### 400 HTTP Error (Bad Request)

This is a common error indicating that the information you sent to the server, the website address perhaps, is not able to be processed for one of many reasons. The error message will be seen within the browser window, and will occur within any operating system.

**What to do:** there are a number of reasons why a 400 error will be recorded, so the routine is to perform one of the following:

a) Most often the 400 error is displayed thanks to incorrectly typed website addresses, or malfunctioning links. Check the spelling of the address carefully and look for characters that are not typically present in addresses.

b) Perform a clear on your cookies; this is most likely to work should your 400 error refer to a Google request. Cookies can be corrupt or out of date.

c) DNS records stored on your computer may be out of date, so clearing your DNS cache may solve the problem. Use the ipconfig/flushdns prompt in Command Prompt on Windows.

d) Your browser cache may contain corrupt versions of the web page required, hence could be the cause of the 400 error. A quick and simple clearing of the cache could solve the problem.

e) It is not common but a 400 error could mistakenly refer to a gateway timeout, in which servers take too long to respond.

f) Should the above not be helpful you need to contact the website concerned. Sometimes a 400 error is at their end of the chain, and not yours. Send them an email informing them of the problem.

g) The security of your device should always be kept updated; make sure you install all updates from Microsoft, and invest in anti-virus software as a matter of course. Poor security can lead to problems that relate to 400 errors.

h) If none of the above help it could be that you need to come back in a while and try again; the problem is with the website, so they will fix it in time.

### 401 HTTP Error (Unauthorized)

This commonly seen code tells you that the resource you were requesting requires a password and user ID. You have either not logged in with those, or you have given incorrect information. You will see it within the browser window, and on any operating system.

**What to do:** there are several possible reasons for a 401 error:

a) You may have entered an incorrect URL or you could have been trying to access a part of the site reserved for authorized users only. Check the URL you have entered is correct.

b) Search the website’s home page for links to ‘Login’ or ‘Secure Access’ and enter your password and user name as requested. Should you now already have those, there will be instructions on how to register.

c) Should you believe that such authorization is not necessary them you may be seeing a 401 message mistakenly. The answer is to contact the website directly and tell them the details of your problem, for others may be having the same problem. It’s worth knowing that the webmaster may usually be found at webmaster@’insertwebsitename.com’.

d) If you get the message after logging in it is possible your information is incorrect. There will be directions on the website to reset your information.

### 402 HTTP Error (Payment Required)

An as yet unused code that is reserved for possible use in the future.

### 403 HTTP Error (Forbidden)

This strict message says that the information you are trying to reach is forbidden to you. As above, there are many reasons why you might see a 403 Forbidden message.

**What to do:** The following are several steps you can take should you receive this message:

a) First, make sure the URL is correct and that the page you are looking for is an actual page; do not get confused by directory addresses, which are likely to be forbidden in most websites and will naturally return a 403 when requested.

b) It could be that a previously cached version of the page you are looking for is the problem; clear the web browser cache.

c) If you are logged into a site and are seeing a 403 when requesting a page, it is probable that you do not have the authority to access that particular information.

d) Cookies, out of date or corrupt, can also lead to a 403; clear your browser cookies and try again.

e) The 403 could be as a result of problems at the website, so let them know of the problem via email.

f) Sometimes IP addresses – and occasionally ISP’s – are blacklisted by websites; this could be a cause of a 403, so check with the website concerned.

g) Try again later, especially if you are aware that the 403 error is not just occurring for you. It could be they are aware of the problem and working to fix it.

### 404 HTTP Error (Not Found)

A 404 error tells you that the page requested simply could not be found on the website’s server. It could be that you are trying to find a page that has been removed, or it might be that the website has failed to inform you of a redirect or has not performed it correctly. A 404 will occur if this is the case.

**What to do:** there are several different possible solutions to a 404 error:

a) Use the F5 function to retry loading the page, or retype the URL in the address bar and try again. Sometimes refreshing the page can eliminate a 404 message as there is not actually a problem.

b) Make sure you have typed the URL correctly as incorrect spelling or unnecessary characters can result in a 404.

c) Move up through the directories by eliminating the sections in the website address: i.e. if you began with www.website.com/x/y/z then try www.website.com/x/y and so on, until something loads. If all that can be loaded is the Home page, try and use a search function to find the resource you require. If there is no search function, use navigation by the links on the home page.

d) Use a search engine and you may find that you are entering the wrong URL; once you find the correct one, be sure to bookmark it so that you do not get repeated 404 errors.

e) If you find that you can reach the resource on one device but not on another it could be that your browser cache on the troubled device is causing the problem. Cookies can also have the same effect, so clear them both on the problematic device.

f) You could try using different DNS servers to access the website, especially if the entire website is returning a 404, which is not a common occurrence. Sometimes censorship can return a 404, in other words you may be trying to access restricted content, and a DNS server change may help.

g) If none of the above fixes the 404 let the website known by email or other forms of contact, as they may not be aware that the error is present. They will be pleased to receive the information and will set about fixing it.

### 405 HTTP Error (Method Not Allowed)

However you are trying to access the information you require, it is not permitted. You may be using a GET method where POST is required, so you need to check the request method. The response should include a list of valid methods.

**What to do:** check the script you are trying to run is supported by the ISP, as some do not allow them to be run. This will return a 405. It may also be that the form you are trying to use is not permitted; for processing forms, some methods are not supported by certain ISP’s and will return a 405. Your ISP should have the information required, or you can contact the website direct.

### 406 HTTP Error (Not Acceptable)

When a user makes a request to a browser, the browser then files that request to the server; a 406 error says that the format used is not acceptable to the server, so the information returned is the error message. This error can also result from certain firewall applications, which look for set rules that need to be adhered to.

**What to do:** if the problem is with said firewall – known as Mod\_security – the system can be disabled. If you are technically minded and have access to the source codes and data streams, plus the Access Headers, you may be able to deduce the cause of the 406, but it is probably best to contact the technical support team responsible for the system concerned.

### 407 HTTP Error (Proxy Authentication Required)

Although the information sent to the web server is correct, the system has detected that to process the information requires the use of a proxy, which will require a password and ID for authentication.

**What to do:** your ISP should be able to provide you with the relevant information on available proxy servers, or you could try navigating to the resource by another method, perhaps an alternative URL for the proxy server list. Contact the relevant parties if problems persist.

### 408 HTTP Error (Request Timeout)

When a web server detects that an inordinate length of time has passed between either a connection to a web browser or a socket, it will ‘drop’ the connection and return a 408 error message. As the request has genuinely been timed out, the user must retry in order to connect. There are set time periods that website servers will wait for a connection, and a 408 will be the result should this be exceeded.

**What to do:** there are number of factors that can influence a 408, hence there are several solutions to the problem:

a) Refresh the website or re-enter the URL; slow connections are not unusual but are generally temporary. Be careful of doing this many times when using online shopping carts; it has been known to create several orders by mistake, and hence take many payments. It is to be noted that online merchants do tend to have protection against this installed.

b) It could be that it is your internet connection at fault; if you suspect this, try an alternative website and see if it loads. If the other page loads as normal, it’s likely the issue is not with you, but with the website concerned. If others are slow you could run a speed test and see if there is a problem, or contact your ISP to see if they are experiencing problems.

c) 408’s can occur when a website receives an unexpected increase in traffic. This can give the servers a hard time and result in problems. Once traffic dies down you may find the site loads as normal, so try again later.

d) If you continue to have problems, or are aware that others are also finding a 408, it is advisable to contact the website and let them know the situation, as they will want to rectify it as soon as possible.

### 409 HTTP Error (Conflict)

The information you are trying to request has resulted in a conflict, hence the 409 error. For instance, if you try and upload something that is older than that in existence you will create a conflict in version control. Also, if you are trying to perform an action that is beyond your granted authority it may generate a 409 error.

**What to do:** you must contact your ISP and find out why the conflict is occurring, either by email or by other means, as only they can resolve the problem.

### 410 HTTP Error (Gone)

This very definite error code is returned when you try to access a page or site that is no longer available, or is deliberately hidden from view. Such pages should have been removed from the index by search engines, but may have been overlooked. This is a permanent situation – the resource is simply not there anymore, and no alternative URL has been given for it.

**What to do:** there is pretty much nothing you can do about a 410 error; as the page or site is no longer there, and there is no redirection address, it cannot be accessed. If you believe the URL should be live then you may try and access it from a different angle, but you will most likely see another 410.

### 411 HTTP Error (Length Required)

This HTTP error is usually only applicable when you are trying to add data to the web server rather than find it. All such transactions will require a length to be specified, and a header will be offered in which to define it. If the length is not defined, the system will return a 411 error.

**What to do:** it is best to contact the ISP and ask them for the relevant information, unless you are technically adept enough to work out what the system is having a problem with.

### 412 HTTP Error (Precondition Failed)

All requests to servers contain certain conditions specified within the Request Header field attached to the request; a 412 error indicates that something within these parameters was not within the conditions required, and hence has been rejected. A 412 can also be returned in the event of suspicious or bad request parameters.

**What to do:** in the first instance, try again using an alternative browser as this often works. Also, if the website is under your control, turn off such as mod\_security, or alter the set rules. If this is not the solution, talk to your ISP to deduce why you ate getting a 412 error.

### 413 HTTP Error (Request Entity Too Large)

A 413 error simply indicates that the information you are trying to send to the server was simply too big; perhaps you were trying to upload a large file or similar. The server will have set parameters as to what is considered too large.

**What to do:** it is recommended you contact your ISP for information on the limitations with regard to 413 errors.

### 414 HTTP Error (Request URL Too Long)

URL’s are generally expected to be within a set length, defined in characters, and can sometimes be decreed as too long. Should this be the case you will see a 414 error. You should be able to find out the maximum length of a URL set by your web server, and it is helpful to keep it as short as possible in order to help with the processing of information.

**What to do:** hyperlinks are often lengthy and convoluted, and can constitute the problem or a 414 error; clearing your cache may help the situation, but you may need to contact your ISP and find out exactly why the system is rejecting the URL and if it is actually valid.

### 415 HTTP Error (Unsupported Media Type)

This code implies that part of the request to the server included a format not supported; it could be an image or another media file o a type that the server cannot deal with.

**What to do:** the best solution is to find out from your ISP the media formats that are supported and try to make the request using one of these.

## 500 Series – Server Error

### 500 HTTP Error (Internal Server Error)

This is one of the more commonly seen error messages as it covers a range of problems; it simply means that there is a problem on the server relating to the website, but is not specific about it. In other words, it’s not your problem, it’s theirs, and it could be down to a programming fault.

**What to do:** as 500 is a very general code there are several things that could cause the problem and, therefore, a number of possible solutions:

a) Try refreshing and reloading the site by either the refresh, F5 or typing in methods, as the problem could just be temporary. Be careful when doing this while checking out on a shopping site as, although there are usually security provisions in place, it can result in multiple purchases.

b) A cached version of the page you are trying to view could be the cause of the 500 error, although it is not a common problem. However, it is worth clearing the cache as it has been known to relieve the problem, and it is quick and simple.

c) Cookies may also be the problem, so again, deleting cookies on your browser can be a quick and effective way of alleviating a 500 error.

d) Sometimes a server will return a 500 error when the problem is actually a 504 Gateway Timeout error (see below); try the advice there to see if it helps.

e) If none of the above work then it may be advisable to let the relevant persons at the website know, as they may be unaware of the problem and will wish to fix it.

f) Try again later, as the 500 error may be the result of a temporary problem already known to them.

### 501 HTTP Error (Not Implemented)

This message tells you that the method you are using to request the information cannot be processed by the server, either because it doesn’t understand it, or has not been told how to do it.

**What to do:** you need to make sure you are using a request format that is valid, or the server itself may be outdated and need updating.

### 502 HTTP Error (Bad Gateway Errors)

The 502 error also covers a lot of possibilities, and informs you that a server needed to perform the action requested has returned a response that was not valid to the operation.

**What to do:** there are many possible reasons for a 502 error, and many possible solutions to the problem:

Note that a 502 error is, more often than not, a problem experienced by two servers trying to communicate with each other; this means it is not a problem with your system, or with your connection. It could be that something you are doing is influencing the problem, so it is best to try the following just in case.

a) Reload the URL once more, either by refreshing, using 5 or by typing the URL again; it is entirely possible that the problem is temporary and short term, and this first step often works.

b) Close your browsers and open a new one, and now try to load the website. If the problem was on your part, or that of your browser, this simple act should solve the problem in an instant.

c) Old or corrupt files that are cached on your system can produce a 502 error in some circumstances, so emptying your cache may be the simple answer to the problem.

d) Also, cookies may be the problem, as above; clear cookies and you may have the answer. You can clear those that are specific to the site concerned if you wish not to clear them all.

e) Use the Safe Mode option; by starting your browser in Safe Mode you are doing so without any other add-ons that may be having an effect on the 502 error. If this solves the problem you have an indication that it is an extension or otherwise relating to your browser that is at fault. Search through the settings until you find the setting that is causing the problems. Please note, this is not the same as Windows Safe Mode, it is that specific to your browser.

f) You may want to try a different browser if the problem persists; Chrome, Firefox, Internet Explorer and Safari are among the most popular, and you might find that a different browser, for certain reasons, overcomes the 502 error.

g) The old restart trick might also help; sometimes rebooting your machine will remove the 502 error, as the problem could have been due to your network connection. This will possibly be the case if you are seeing a 502 on more than one site.

h) Further to the above, rebooting your router, modem or any other devices on the network may solve a 502, as they could be causing problems in the way they are connected. Make sure that, when you turn them back on, you begin with the modem, router, and then any devices between that and your computer.

i) Some issued with DNS servers can cause 502 errors, so it may be worth changing to a different DNS server. These are assigned to you by your ISP, who may be able to help you if you are not sure what to do.

j) If problems persist it may be helpful to contact the relevant persons at the website as they may not know about the 502 error, or may already be working on rectifying the problem.

k) If everything in your home network is operational and the website personnel cannot see a 502, you should contact your ISP as there may be an issue with their network could be the cause of the problem.

l) As with all things, try again later on; if the website administrators are aware of the problem they will be trying their best to fix it, and will do so in good time. Remember, it is more than likely that a 502 error is not your fault, but is a problem with the website’s network or that of your ISP provider, but trying the above will do no harm at all.

### 503 HTTP Error (Service Unavailable)

This often-seen error message carries a simple message: the server for the website you are trying to access is not available at that moment. It could be due to heavy traffic or maintenance.

**What to do:** there are a number of reasons why you might see a 503 error, hence the following solutions may apply:

a) Try the website again by either refreshing, pressing F5 or typing the address once more, as it is entirely possible that the 503 error is a temporary one and, in many cases, this will solve the problem. Take care if doing this when making a purchase as, although security measures are usually in place, it can result in multiple purchases.

b) Reboot your router and then your computer; this is recommended especially if the error message reads ‘DNS failure’. It is still the greater likelihood that the problem lies with the website’s servers, but in some cases it could be one of your devices that is malfunctioning and this could solve the problem.

c) Contact the website and let them know of the problem; they may already be aware but if not they will welcome your message and begin to rectify the problem.

d) Try again later on; this is probably the best of all options because, as the problem is out of your control, those responsible for it will likely be working on it and after a while the service will likely return to normal. It could also be that the 503 is down to an unexpected influx of traffic, in which case you will be able to access the site when they leave.

### 504 HTTP Error (Gateway Timeout Error)

This common error message tells you that the server your request is trying to communicate with did not respond in good time, most likely due to maintenance or a fault.

**What to do:** there are many possible causes of a Gateway Timeout Error, hence the following solutions may be applicable:

a) Try refreshing and reloading the site by either the refresh, F5 or typing in methods, as the problem could just be temporary. Be careful when doing this while checking out on a shopping site as, although there are usually security provisions in place, it can result in multiple purchases.

b) Further to the above, rebooting your router, modem or any other devices on the network may solve a 504, as they could be causing problems in the way they are connected. Make sure that, when you turn them back on, you begin with the modem, router, and then any devices between that and your computer.

c) Some issued with DNS servers can cause 504 errors, so it may be worth changing to a different DNS server. These are assigned to you by your ISP, who may be able to help you if you are not sure what to do.

d) Contact the website and let them know of the problem; they may already be aware but if not they will welcome your message and begin to rectify the problem.

e) If everything in your home network is operational and the website personnel cannot see a 504, you should contact your ISP as there may be an issue with their network could be the cause of the problem.

f) Try again later on; this is probably the best of all options because, as the problem is out of your control, those responsible for it will likely be working on it and after a while the service will likely return to normal.  
505 HTTP Error (HTTP Version Not Supported) – the website you are trying to access does not support the HTTP protocol that you are currently using, which is commonly HTTP/1.1.

**What to do:** you need to upgrade your web server software, in order to be able to access the website and others that may also return a 505 error.

Read more: <https://www.serped.com/most-common-http-errors-explained-and-how-to-fix-them/15#ixzz5ubKMcFcd>

### Safe Methods

As per HTTP specification, the **GET and HEAD methods should be used only for retrieval of resource representations** – and they do not update/delete the resource on the server. Both methods are said to be considered “**safe**“.

This allows user agents to represent other methods, such as **POST, PUT and DELETE**, in a special way, so that the user is made aware of the fact that a possibly unsafe action is being requested – and they can **update/delete the resource on server** and so should be used carefully.

### Idempotent Methods

The term idempotent is used more comprehensively to describe an **operation that will produce the same results if executed once or multiple times**. This is a very useful property in many situations, as it means that an operation can be repeated or retried as often as necessary without causing unintended effects. With non-idempotent operations, the algorithm may have to keep track of whether the operation was already performed or not.

In HTTP specification, The methods **GET, HEAD, PUT and DELETE are declared idempotent methods**. Other methods OPTIONS and TRACE SHOULD NOT have side effects so both are also inherently idempotent.

Important Questions and Anseres

**Question 1 : What is REST?**  
Answer : REST is an architectural style of developing web services which take advantage of ubiquity of HTTP protocol and leverages HTTP method to define actions. REST stands for *REpresntational State Transfer*.  
  
  
**Question 2 : What is RESTFul Web Service?**  
Answer : There are two popular way to develop web services, using SOAP (Simple Object Access Protocol) which is XML based way to expose web services and second REST based web services which uses HTTP protocol. Web services developed using REST style is also known as [RESTful Web Services](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Frest-web-services-using-java-ee%2F).  
  
  
Question 3 : What is HTTP Basic Authentication and how it works?  
  
  
**Question 4 : Can you tell me which API can be used to develop RESTFul web service in Java?**  
Answer : There are many framework and libraries out there which helps to develop RESTful web services in Java including JAX-RS which is standard way to develop REST web services. Jersey is one of the popular implementation of JAX-RS which also offers more than specification recommends. Then you also have RESTEasy, RESTlet and Apache CFX. If you like Scala then you can also use Play framework to develop RESTful web services.  
  
  
Question 5 : How do you configure RESTFul web service?  
  
Question 6 : How you apply security in RESTFul web services?  
  
Question 7 : Have you used securing RESTful APIs with HTTP Basic Authentication  
  
Question 8 : How you maintain session in RESTful services?  
  
  
**Question 9 : Have you used Jersey API to develop RESTful services in Java?**  
Answer : Jersey is one of the most popular framework and API to develop REST based web services in Java. Since many organization uses Jersey they check if candidate has used it before or not. It's simple to answer, say Yes if you have really used and No, if you have not. In case of No, you should also mention which framework you have used for developing RESTful web services e.g. Apache CFX, Play or Restlet.  
  
  
Question 10 : How you test RESTful web services?  
  
Question 11 : What is WADL in RESTFul?  
  
  
**Question 12 : What do you understand by payload in RESTFul?**  
Answer : Payload means data which passed inside request body also payload is not request parameters. So only you can do payload in POST  and not in GET and DELTE method  
  
  
**Question 13 : Can you do payload in GET method?**  
Answer : No, payload can only be passed using POST method.  
  
  
**Question 14 : Can you do payload in HTTP DELETE?**  
Answer : This is again similar to previous REST interview question, answer is No. You can only pass payload using HTTP POST method.  
  
  
**Question 15 : How much maximum pay load you could do in POST method?**  
Answer : If you remember [difference between GET and POST request](http://java67.blogspot.sg/2014/08/difference-between-post-and-get-request.html) then you know that unlike GET which passes data on URL and thus limited by maximum URL length, POST has no such limit. So, theoretically you can pass unlimited data as payload to POST method but you need to take practical things into account e.g. sending POST with large payload will consume more bandwidth, take more time and present performance challenge to your server.  
  
  
**Question 16 : What is difference between SOAP and RESTFul web services?**  
Answer : There are many difference between these two style of web services e.g. SOAP take more bandwidth because of heavy weight XML based protocol but REST takes less bandwidth because of popular use of JSON as message protocol and leveraging HTTP method to define action. This also means that REST is faster than SOAP based web services. You can derive many differences between SOAP and RESTful with the fact that its HTTP based e.g. REST URLs can be cached or bookmarked. Here are few more differences between them :

[](https://pluralsight.pxf.io/c/1193463/424552/7490?u=https://www.pluralsight.com/courses/restful-services-java-using-jersey)

**Question 17 : If you have to develop web services which one you will choose SOAP OR RESTful and why?**  
Answer : You can answer this question based upon your experience but the key here is if you know difference between them than you can answer this question in more detail. For example, its easy to develop RESTful web services than SOAP based web services but later comes with some in-built security features.  
  
  
**Question 18 :  What framework you had used to develop RESTFul services?**  
Answer : This is really experience based question. If you have used Jersey to develop RESTFul web services then answer as Jersey but expect some follow-up question on Jersey. Similarly if you have used Apache CFX or Restlet then answer them accordingly.  
  
  
That's all in this list of some good **RESTful web service interview questions for Java developers**. Though this list is meant for Java developer, you can use this questions to check any candidate's knowledge on REST style web services independent of programming language because REST doesn't say that you need to implement web service in Java only. Since it take advantage of ubiquitous HTTP protocol you can build backed with any web technology stack e.g. Java, .NET or any other.  
  
Read more: <http://www.java67.com/2015/09/top-10-restful-web-service-interview-questions-answers.html#ixzz5WEs0QwMp>

interesting *tricky or tough Java web service question* asked in Interviews then please share with us :

## **Frequently asked Java webservice interview questions**

[Java Web Service Interview Question and Answers SOAP REST](http://www.shareasale.com/m-pr.cfm?merchantID=53701&userID=880419&productID=546412145)Here is my list of frequently asked interview question on Java web service in any [Core Java Interview](http://java67.blogspot.sg/2012/08/10-java-coding-interview-questions-and.html). As SOAP is an standard way of making web service call which uses XML, good knowledge of XML and Java is expected from you and Interviewer may ask some*XML interview question*s as well.

What is Web Service ?

What is SOAP ?

What is REST Web Service ?

What is difference between REST Web Service and SOAP web service ?

Can a Java client can talk to C++ Server using Web Service ?

What is WSDL?

What is UDDI?

Does Web Service call is synchronous or asynchronous ?

How do you handle errors in Web Service call ?

What is JAX-RPC ?

Have you worked on Spring and Web-services ?

What is WebServiceTemplate etc

What is difference between RMI and Web Services

Read more: <http://www.java67.com/2012/09/top-10-java-web-service-interview-question-answer-soap-rest.html#ixzz5WEs5dwqx>

What is Web Service ?

What is SOAP ?

What is REST Web Service ?

What is difference between REST Web Service and SOAP web service ?

Can a Java client can talk to C++ Server using Web Service ?

What is WSDL?

What is UDDI?

Does Web Service call is synchronous or asynchronous ?

How do you handle errors in Web Service call ?

What is JAX-RPC ?

Have you worked on Spring and Web-services ?

What is WebServiceTemplate etc

What is difference between RMI and Web Services

2. What are the advantages of Web Services?

3. What are different types of Web Services?

5. What are advantages of SOAP Web Services?

6. What are disadvantages of SOAP Web Services?

7. What is WSDL?

8. What are different components of WSDL?

9. What is UDDI?

10. What is difference between Top Down and Bottom Up approach in SOAP Web Services?

11. What is REST Web Services?

12. What are advantages of REST web services?

13. What are disadvantages of REST web services?

14. What is a Resource in Restful web services?

15. What are different HTTP Methods supported in Restful Web Services?

16. Compare SOAP and REST web services?

17. What are different ways to test web services?

18. Can we maintain user session in web services?

19. What is difference between SOA and Web Services?

20. What is the use of Accept and Content-Type Headers in HTTP Request?

21. How would you choose between SOAP and REST web services?

22. What is JAX-WS API?

23. Name some frameworks in Java to implement SOAP web services?

24. Name important annotations used in JAX-WS API?

25. What is use of javax.xml.ws.Endpoint class?

26. What is the difference between RPC Style and Document Style SOAP web Services?

27. How to get WSDL file of a SOAP web service?

28. What is sun-jaxws.xml file?

29. What is JAX-RS API?

30. Name some implementations of JAX-RS API?

31. What is wsimport utility?

32. Name important annotations used in JAX-RS API?

33. What is the use of @XmlRootElement annotation?

34. How to set different status code in HTTP response?