***Top 20 REST and Spring MVC Interview Questions for Java Developers***

*Since the*[*Spring Framework*](http://javarevisited.blogspot.sg/2018/01/how-to-learn-spring-core-spring-mvc-boot-security-framework.html#axzz55IgfKjy8)*is the most popular and standard framework for developing Java web applications and RESTful web services, a good knowledge of Spring core and*[*Spring MVC*](http://javarevisited.blogspot.sg/2018/01/how-to-learn-spring-core-spring-mvc-boot-security-framework.html#axzz55IgfKjy8)*is expected from any senior Java developer. But, if the job description mentions REST and web services, you also need to be aware of how to develop RESTful web services using the Spring Framework.*

*From Spring 3.1, the framework has been enhanced a lot to support many features needed for the RESTFul API. The****HTTPMessageConverter****can convert your HTTP response to*[*JSON*](http://javarevisited.blogspot.sg/2018/02/how-to-parse-json-with-date-field-in-java-jackson-example.html#axzz56WGunSwy)*or*[*XML*](http://javarevisited.blogspot.sg/2013/01/jaxb-xml-binding-tutorial-marshalling-unmarshalling-java-object-xml.html#axzz56WGunSwy)*just by detecting a relevant library in the classpath, e.g.*[*Jackson*](http://javarevisited.blogspot.sg/2018/02/how-to-parse-json-with-date-field-in-java-jackson-example.html)*and*[*JAXB*](http://javarevisited.blogspot.sg/2013/01/jaxb-xml-binding-tutorial-marshalling-unmarshalling-java-object-xml.html)*.*

*Spring also provides customized annotations for RESTful Web Services, e.g.****@RestController,****which can make your Controller REST more aware, so that you don't need to do common stuff required by every single REST API, like converting the response to JSON.*

*A good knowledge of Spring Security is also mandatory for developing security for RESTful web services in the real world. Since you cannot make life a non-trivial REST API without security, a good knowledge of security basics,*[*HTTP basic authentication*](http://javarevisited.blogspot.sg/2018/01/how-http-basic-authentication-works-in.html)*, digest authentication, OAuth, and JWT is very important.*

## ***20 Spring REST Web Service Interview Questions***

*Here are a couple of frequently asked questions about using REST web services in the Spring Framework.*

***What does REST stand for?***

*REST stands for the REpresentational State Transfer, which uses the HTTP protocol to send data from the server to the client, e.g. a book in the server can be delivered to the client using JSON or XML.*

*However, if you are not familiar with REST, I suggest you to first check out the*[*REST API design and development*](http://bit.ly/2zIGzWK)*to better understand it.*

***What is a resource?***

*A resource is how data is represented in the REST architecture. By exposing entities as the resource, it allows a client to read, write, modify, and create resources using HTTP methods, for example,*[*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?***

*REST API uses HTTP methods to perform operations. Some of the HTTP operations, which doesn't modify the resource at the server, are known as safe operations, including 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?***

*There are some* ***HTTP methods — like GET — that produce the same response no matter how many times you use them,*** *sending multiple GET request to the same*[*URI*](http://www.java67.com/2013/01/difference-between-url-uri-and-urn.html)*will result in same response without any side-effect. Hence, this is known as idempotent.*

*On the other hand, the*[*POST is not idempotent*](http://javarevisited.blogspot.sg/2016/05/what-are-idempotent-and-safe-methods-of-HTTP-and-REST.html)*, 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 requests can be used to update a resource on a server and will give the same end result.*

*You can take a*[*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?***

*Yes,*[*REST*](http://javarevisited.blogspot.sg/2015/08/difference-between-soap-and-restfull-webservice-java.html)*is scalable and interoperable. It doesn't mandate a specific choice of technology either at client or server end. You can use*[*Java*](http://javarevisited.blogspot.sg/2017/11/top-5-free-java-courses-for-beginners.html)*,*[*C++*](http://www.java67.com/2018/02/5-free-cpp-courses-to-learn-programming.html)*,*[*Python*](http://www.java67.com/2018/02/5-free-python-online-courses-for-beginners.html)*, or*[*JavaScript*](http://www.java67.com/2018/04/top-5-free-javascript-courses-to-learn.html)*to create RESTful web services and consume them at the client end.*

*I suggest you read a good book on REST API, like*[*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?***

*The RestTemplate class is an implementation of*[*the Template method pattern*](http://www.java67.com/2012/09/top-10-java-design-pattern-interview-question-answer.html)*in the Spring framework. Similar to other popular template classes, like the 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 RestTemplate example.*

***Which HTTP methods does REST use?***

*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*](http://javarevisited.blogspot.sg/2016/04/what-is-purpose-of-http-request-types-in-RESTful-web-service.html#axzz56WGunSwy)*, and DELETE for removing a resource from the server.*

***What is an HttpMessageConverter in Spring REST?***

* *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 responses to various formats, for example, 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 that the client is expecting.*
* *It will then try to find a registered*[*HTTPMessageConverter*](http://courses.baeldung.com/p/rest-with-spring-the-master-class?affcode=22136_bkwjs9xa)*that is capable of handling that specific content-type and use it to convert the response into that format before sending it to the client.*

***How to create a custom implementation of the HttpMessageConverter to support a new type of request/responses?***

*You just need to create an implementation of the AbstractHttpMessageConverter and register it using the WebMvcConfigurerAdapter#extendMessageConverters() method with the classes that generate a new type of request/response.*

***Is REST normally stateless?***

*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 to process it.* ***It should not rely on previous or next requests or some data maintained at the server end, like sessions.*** *The REST specification puts a constraint to make it stateless, and you should keep that in mind while designing your REST API.*

***What does @RequestMapping annotation do?***

*The @RequestMapping annotation is* ***used to map web requests to Spring Controller methods****. You can map a request based upon HTTP methods, e.g. GET, POST, and various other parameters.*

***For example, if you are developing a RESTful web service using Spring, then you can use, produce, and consume property along with media type annotations to indicate that this method is only used to produce or consume JSON****, as shown below:*

**

*Similarly, you can create other handler methods to produce JSON or XML.*

*If you are not familiar with these annotations, then I suggest you join this*[***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 the basics.*

***Is @Controller a stereotype? Is @RestController a stereotype?***

*Yes, both****@Controller and @RestController are stereotypes****. The @Controller is actually a specialization of Spring's @Component stereotype annotation. This means that the class annotated with the @Controller will also be automatically detected by the* ***spring container****, as part of the container's component scanning process.*

*And, the @RestController is a specialization of the @Controller for the RESTful web service. It not only combines the*[*@ResponseBody*](http://courses.baeldung.com/p/rest-with-spring-the-master-class?affcode=22136_bkwjs9xa)*and @Controller  annotations, but it also gives more meaning to your controller class to clearly indicate that it deals with RESTful requests.*

*Your 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?***

*There are many differences between the @Controller   and @RestController annotations,  as discussed in my earlier article (see the answer for more!), but the most important one is that with the @RestController  you get the @ResponseBody annotation automatically, which means you don't need to separately annotate your handler methods with the @ResponseBody annotation.*

*This makes the development of RESTful web services easier using Spring.*

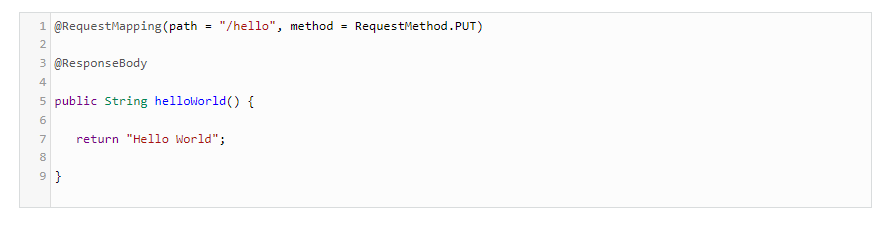
*You can see here to learn more about*[*Spring Boot*](http://bit.ly/2yjLysL)*and how it can help you to create Spring MVC based web applications.*

**

***When do you need @ResponseBody annotation in Spring MVC?***

*The @ResponseBody annotation can be put on a method to indicate 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:*

**

*Alternatively, you can also use the*[*@RestController*](http://courses.baeldung.com/p/rest-with-spring-the-master-class?affcode=22136_bkwjs9xa)*annotation instead of the @Controller annotation. This will remove the need for using @ResponseBody because, as it comes automatically with the @RestController annotation.*

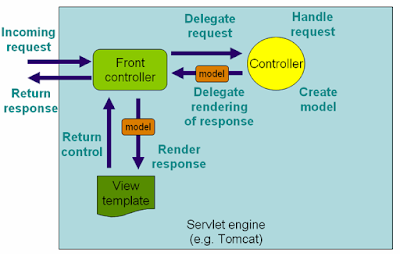
***What does @PathVariable do in Spring MVC? Why it's useful in REST with Spring?***

*This is one of the useful annotations from Spring MVC that allows you to read values from the 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 the URI.*

*This question is normally asked by experienced Spring MVC developers with 4 to 6 years of experience.*

*For example, this*[*URL*](http://myapp.com/books/101)*can be helpful if you want to learn how to extract the id, then you can use the @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.*

**

***What is the HTTP status return code for a successful DELETE statement?***

*There is no strict rule about what status code your REST API should return to after a successful DELETE.* ***It can return 200 Ok or 204 No Content.***

*In general, if the DELETE operation is successful, 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?***

*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.*

***Where do you need @EnableWebMVC?***

*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 an XML configuration.*

*It enables support for the @Controller-annotated classes that use @RequestMapping to map incoming requests to handler methods.*

***When do you need @ResponseStatus annotation in Spring MVC?***

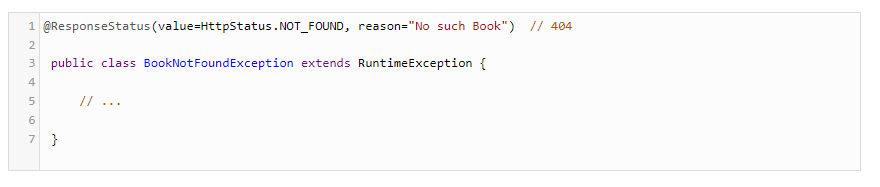
***The @ResponseStatus annotation is required during error handling in***[***Spring MVC***](http://javarevisited.blogspot.sg/2016/12/top-5-spring-and-hibernate-training-courses-java-jee-programmers.html#axzz56WXxxAC0)***and REST.*** *Normally, when an error or exception is thrown at the server side, the web server returns 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 the proper status code, like 404, if the resource is not found. That's where you can use the****@ResponseStatus  annotation, which allows you to send custom HTTP status codes along with proper error message in case of an exception.***

*In order to use it, you can create custom exceptions and annotate them using the @ResponseStatus annotation and proper HTTP status code and reason.*

*When such exceptions are thrown from the controller's handler methods and not handled anywhere else, then the appropriate HTTP response with the proper HTTP status code is sent to the client.*

*For example, if you are writing a*[*RESTful web service*](http://www.java67.com/2015/09/top-10-restful-web-service-interview-questions-answers.html)*for a library that provides book information, then you can use @ResponseStatus to create an exception that returns the HTTP response code 404 when a book is not found instead of the Internal Server Error (500), as shown below:*

**

*If this exception is thrown from any handler method, then the HTTP error code 404 with the reason "No such Book" will be returned to the client.*

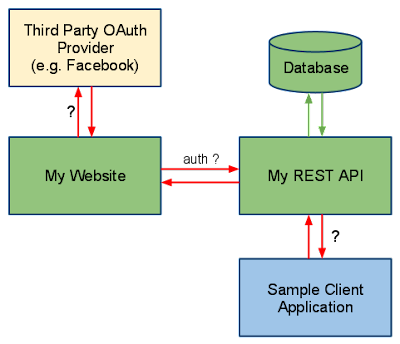
*If you are not familiar with the basics concepts of Spring MVC, Security, and REST, I suggest you go through these*[*REST with Spring*](http://courses.baeldung.com/p/rest-with-spring-the-master-class?affcode=22136_bkwjs9xa)*and*[*Learn Spring Security*](https://courses.baeldung.com/p/learn-spring-security-the-master-class?utm_source=javarevisited&utm_medium=web&utm_campaign=lss&affcode=22136_bkwjs9xa)*courses to gain some experience before your next job interview. These two courses are specially designed to provide you with some real-world experience to boost both your knowledge and experience with Spring MVC, REST, and Spring Security.*

***Is REST secure? What can you do to secure it?***

*This question is mostly asked by experienced Java programmers with about 2 to 5 years of 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 that are 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 the 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.*

**

***Does REST work with transport layer security (TLS)?***

***Transport Layer Security*** *(TLS) is used for* ***secure communication between the 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, in REST, it is up to the server to implement security protocols.* ***The 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 learn more about how to enable SSL in Tomcat.*

***Do you need Spring MVC in your classpath for developing RESTful Web Service?***

*The short answer is:****yes —****you need Spring MVC in your Java application's classpath to develop RESTful web services using the Spring framework.*

*It's actually Spring MVC that provides all useful annotations, like  @RestController,  @ResponseCode ,  @ResponseBody,  @RequestBody, and @PathVariable (see*[*REST with Spring*](http://courses.baeldung.com/p/rest-with-spring-the-master-class?affcode=22136_bkwjs9xa)*). Hence, you must use spring-mvc.jar or the appropriate Maven entry in your pom.xml*

### ***What does it mean for an API to be stateless?***

***Answer:******Statelessness is one of the key principles of REST architecture****. In stateless communication, the server does not store any information about previous communications. In other words, the client and server do not know each other’s state.* ***Every request and response is a new interaction,*** *and each request includes everything the server needs to give a successful response.*

***Statelessness simplifies client-server interactions*** *because the server does not rely on past requests to process future requests, and thus* ***does not need to consume space and resources storing data from these requests****.*

### ***Which protocol do REST APIs use?***

***Answer:****REST APIs use the HTTP protocol to communicate with clients. This allows REST APIs to be easily deployed over the internet, since HTTP is the same protocol that is used to deliver web pages to client browsers.*

### ***Which markup languages are primarily used to represent resources in REST APIs?***

***Answer:****In REST APIs, XML (extensible markup language) and JSON (JavaScript Object Notation) are the two most common languages for representing resources.*

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

*Answer: POST and PUT are similar, but not exactly the same. POST is for creating a resource on the server, whereas PUT is for replacing a resource at a specific URI with another resource****. If you use PUT at a URI that already has an associated resource, PUT will replace that resource. If there is no resource at the specified URI, PUT creates one.***

*Additionally,* ***PUT is idempotent****, which means that calling it multiple times will only result in one resource.* ***This is because each call replaces the existing resource (or creates a new one if there is nothing to replace).***

***POST is not idempotent.*** *If you call POST 10 times, you’ll end up with 10 different resources on the server, each with its own URI.* ***This also means that POST responses are cacheable, whereas PUT responses are not.***

### ***What is CRUD?***

***Answer:****CRUD stands for “Create, Read, Update, and Delete.” These are the four basic actions that can be performed on databases through a REST API. Each action corresponds to an HTTP request method:*

* *Create = POST*
* *Read = GET*
* *Update = PUT*
* *Delete = DELETE*

### ***What is messaging in the context of REST?***

***Answer:****In REST,* ***messaging refers to the back-and-forth communication between the client and API****. An interaction always starts with the client messaging the API with an HTTP request. The API processes this request, then sends back an HTTP response that gives the status of the request and any resources the client asked for.*

### ***What are the main parts of an HTTP request?***

***Answer:******HTTP requests are sent by the client to the API****. They request data or perform some action on the server. There are* ***five main components*** *of an HTTP request in REST:*

* ***Start line:****Indicates the intended action of the request and includes:*
* *A****request method****that indicates the HTTP request method to be performed on the resource (i.e., GET, POST, PUT, DELETE).*
* *A****URI****that identifies the requested resource on the server.*
* *The****HTTP version****being used, which signals which version the API should respond with.*
* ***HTTP Request Header:****Lists* ***metadata about the request****, such as the user agent, file formats the client will accept, format of the request body, language, caching preferences, etc.*
* ***HTTP Request body:****Contains any data associated with the request.* ***This is only necessary if the request is to modify data on the server with the POST or PUT methods.***

### ***What are the main parts of an HTTP response?***

***Answer:******HTTP responses are sent by the API to the client****. They inform the client that the requested action was (or was not) completed and to deliver any requested resources. There are four main components of an HTTP response:*

* ***HTTP version:****The version of HTTP version used.*
* ***Status line:****Indicates the status of the request with an* ***HTTP response status code****.*
* ***HTTP Response Header:****Lists* ***metadata about the response****, such as the date, server, user agent, file formats of the returned resources, caching information, etc.*
* ***HTTP Response body:****Contains the resource data that was requested by the client, and is also called the* ***payload****.*

### ***What are some common HTTP response status codes you might see when working with a REST API?***

***Answer:****HTTP response status codes tell the client the result of the requested action (GET, POST, etc.). Some common codes you’ll see in HTTP responses are:*

* ***200 OK:****The request succeeded.*
* ***201 Created:****The request succeeded and a resource was created.*
* ***400 Bad Request:****The request was not fulfilled due to an error in the request, such as a typo or missing data.*
* ***401 Unauthorized:****The request was not fulfilled because the client is not authenticated or authorized to access the requested resource.*
* ***403 Forbidden:****The request was not fulfilled because the client is authenticated, but not authorized to access the requested resource.*
* ***404 Not Found:****The request was not fulfilled because the server could not locate the requested resource.*
* ***405 – METHOD NOT ALLOWED***  
  *If URI has a different http method and if the invoking http method is different. E.g. if /getEmployee has a GET http method and if you are invoking the endpoint with POST http method.*
* ***500 Internal Server Error:****The request was not fulfilled due to an unexpected problem with the server. (See also:*[***500 Internal Server Errors: What They Are & How to Fix Them***](https://blog.hubspot.com/marketing/http-500-internal-server-error)*)*
* ***502 Bad Gateway:****The request was not fulfilled due to an invalid response from an upstream server.*
* ***503 Service Unavailable:****The server was unable to process the request due to maintenance, overloading, or another temporary interference.*

### ***What is a resource?***

***Answer:****In REST, every accessible piece of content on the server is labeled as a resource. A resource is an object with a type, associated data, a relationship with other resources on the server, and a list of methods that can be used with it. For example, a resource could be an HTML or text file, a data file, an image or video, or an executable code file.*

*A resource is identified with a uniform resource identifier, or URI.* ***Clients access resources by including their URIs in HTTP requests.***

### ***15. What is a URI?***

***Answer:****URI stands for uniform resource identifier. In REST, a URI is a string that identifies a resource on a web server. Each resource has its own unique URI which, when included in an HTTP request, allows clients to target that resource and perform actions on it. The process of targeting a resource with its URI is called “addressing.”*

*The format of a URI is as follows:*

*<protocol>://<service-name>/<ResourceType>/<ResourceID>*

### ***16. What is caching?***

***Answer:******Caching is the method of temporarily storing a copy of a server response in a location (like computer memory) in order to retrieve it more quickly in the future.***

*When working with REST APIs, Caching improves server performance by reducing the work the server has to do to fulfill the request. Caching also makes applications that use the API run faster since they don’t need to send a new request every time they need a resource.*

***Cache duration of a resource (i.e., how long the resource can be cached by the client before the resource is retrieved again) is specified in the HTTP response header with the Cache-Control field.***

### ***17. What is payload?***

***Answer:****“Payload”* ***refers to the data in the body of the HTTP request and/or response messages*** *in GET or POST requests.*

*For example, if you request a specific tweet from the Twitter API, the payload comprises the document containing the tweet text and any associated files for rendering the tweet on a page.*

***Payload can also be included in the HTTP request with the POST method****. If you want to post a tweet through Twitter's API, the tweet text that you send in your POST request is the payload.*

### ***18. What’s a real-world example of a REST API?***

***Answer:****Here are some examples of REST APIs in use:*

* ***Twitter allows publishing sites to pull information like tweets, users, tweet streams, and so on.*** *Developers can also write programs to post tweets through the API instead of the website interface.*
* *Airlines expose their flight times and prices through APIs so travel and ticketing sites can use them.*
* *To display weather information, weather apps harness public APIs that share weather data.*
* *Public transportation services usually make their data publicly in real-time via APIs so that mapping and navigation apps (like Google Maps) can use them.*
* *In turn,*[***Google Maps hosts several APIs***](https://blog.hubspot.com/website/google-maps-api)*that make its mapping data available to developers. Developers leverage these APIs to palace dynamic maps on their websites or in their apps.*

### ***19. What is the difference between REST and SOAP?***

***Answer:***[*REST and SOAP*](https://blog.hubspot.com/website/rest-vs-soap)*(Simple Object Access Protocol) are two different approaches to building APIs. Here are the key differences between them:*

* ***SOAP is a strict protocol for building secure APIs****. REST is not a protocol — it is an architectural style dictated by a set of guidelines (see question 3).*
* *REST APIs are simpler to build, more lightweight, and generally faster than SOAP APIs.*
* ***SOAP APIs are considered more secure than REST APIs, though REST APIs can still implement safety features to make them reasonably secure.***
* *REST allows caching of responses, whereas SOAP does not.*
* *SOAP encodes data in XML format. REST allows you to encode data in any format, though XML and JSON are most popular.*

### ***20. What is the difference between REST and AJAX?***

***Answer:******Asynchronous JavaScript, or AJAX, is a set of web development techniques used in web applications****. At its core, AJAX allows a web page to make requests to a server and update the page interface without needing to refresh.*

*An* ***AJAX client*** *might utilize REST APIs with its requests, but AJAX doesn’t have to work with REST APIs exclusively. REST APIs can communicate with any client, whether the client uses AJAX or not.*

*Also, unlike* ***REST which uses HTTP requests and responses for messaging****, AJAX sends its requests to the server with the* ***XMLHttpRequest*** *object that is built into* ***JavaScript****. Server responses are executed by the page’s JavaScript code to alter the page content.*

### ***21. What are some benefits of REST?***

***Answer:****REST is by far the most common method for building web APIs. Here are some reasons why:*

* *REST is based around HTTP and fits within the existing infrastructure of the web, making it easy to implement by web applications.*
* *REST uses simple web technologies like XML and JSON, making it easy to learn.*
* *Because REST communications are* ***stateless,*** *the client and server are decoupled. This means that integrations are scalable and easy to build and manage over time.*
* *The* ***REST architecture is flexible enough to adapt to a huge variety of use cases.***
* ***REST is a lightweight architecture****. Applications built with REST are generally faster than those built with other*[***types of APIs***](https://blog.hubspot.com/website/types-of-apis)*.*
* *REST is easy to test in the browser with an API testing tool.*

### ***22. What are some drawbacks of REST?***

***Answer:****While* ***statelessness is a benefit of REST****, it can sometimes be a disadvantage too. REST does not preserve state. In other words, the server does not keep records of past interactions.* ***If preserving state is necessary, that responsibility falls on the client.***

*Additionally, REST is* ***less strict with its security measures than SOAP****,* ***so developers need to be cautious and only work with APIs from legitimate, reputable providers****. It also makes REST a poor choice for sending confidential information between servers and clients.*

### ***23. How do you test APIs?***

***Answer:****There are many*[*software tools designed for testing RESTful APIs*](https://blog.hubspot.com/website/api-testing-tools)*—* ***Postman****, JMeter, and Katalon Studio are a few. The testing process usually involves sending various requests from your testing tool and monitoring how your API responds.* ***Many testing tools also support automated testing, allowing you to run many different scenarios quickly.***

*To learn more about how API testing works and what kinds of tests you can run, see our*[*guide to API testing*](https://blog.hubspot.com/website/api-testinghttps:/blog.hubspot.com/website/api-testing)*.*

### ***24. How do you keep REST APIs secure?***

***Answer:****REST APIs do not employ as strict security measures as SOAP APIs, and therefore should not be used to send or retrieve sensitive information. However, good REST APIs still implement*[*safety measures*](https://blog.hubspot.com/website/api-security)*for secure and reliable data transfers.*

* ***Authentication and authorization:****All requests to the API should be authenticated and authorized. Authentication is the process of* ***verifying the identity of the client****, and authorization is confirming that the client has permission to access the requested resources.*
* ***Validation:****After authentication and authorization, requests still need to be scanned for potentially malicious code before the API gives access to its resources.* ***Otherwise, a server will be vulnerable to an injection attack.***
* ***Encryption:***[***TLS/SSL encryption***](https://blog.hubspot.com/marketing/what-is-ssl)*secures the connection between client and server and prevents attackers from intercepting requests and responses.* ***TLS – Transport Layer Security***
* ***Rate-limiting:****Rate-limiting methods like quotas and throttling prevent brute-force attacks like*[***DDoS***](https://blog.hubspot.com/website/what-is-ddos-attack)*that attempt to slow or crash the server.*
* ***No sensitive information in URIs:****Protected information (e.g., username, password, or authentication token) should not be visible in the URI of a resource.*

*For more in-depth explanations, see our*[*post on API security*](https://blog.hubspot.com/website/api-security)*.*

### ***25. What are some main characteristics of REST?***

***Answer:****If you’re asked to point out the* ***main features that distinguish REST APIs from others****, here are some points to touch on:*

* ***REST*** *uses* ***the HTTP protocol for communication****.*
* ***REST makes server resources available via URIs****. Each resource has a* ***unique URI.***
* ***REST is stateless****, meaning the server does not store information about past communications with clients.*
* *REST uses* ***GET to retrieve resources from a server****, whereas other web service methods use POST.*

### ***What are the principles of REST?***

***Answer:****REST APIs must adhere to five requirements:*

* ***Client-server decoupling:****The client and server can only interact in a series of requests and responses. Only clients can make requests, and only servers can send responses.* ***This simple principle allows both parties to operate independently of each other.***
* ***Uniform interface:****All communications between the client and server must follow the same protocol. For REST, this protocol is HTTP.* ***A uniform interface simplifies integrations because every application is using the same language to request and send data.***
* ***Stateless:****In stateless communication, the server does not store any information about past requests/responses. Each request and response contains all information needed to complete the interaction.* ***Stateless communication reduces server load, saves memory, and improves performance. It also eliminates the possibility of a failed request caused by missing data.***
* ***Layered system:****Layers are servers that sit between the client and API server. These additional servers perform various functions, like identifying spam and improving performance (See also:*[***What Is a CDN?***](https://blog.hubspot.com/website/what-is-a-cdn-definition)*). In REST,* ***layers are modular*** *and can be added and removed without affecting the messages between the client and the API server.*
* ***Cacheable:****Server responses indicate whether or not the resource is cacheable, so that clients can cache any resources to improve performance.*

### ***1. What is REST?***

***Answer:****REST stands for Representational State Transfer, and is an architectural style based on the Hypertext Transfer Protocol (HTTP) for developing web-based applications.*

*REST outlines several guidelines that web services must follow to be considered RESTful. These guidelines ensure that requests and resources are sent easily and efficiently between client and server using standardized HTTP methods.*

### ***2. What is a REST API?***

***Answer:****An application programming interface is a software-to-software interface that allows otherwise separate applications to interact and share data. For example, a news website could*[*leverage the Twitter API*](https://blog.hubspot.com/website/how-to-use-twitter-api)*to automatically find relevant tweets and include them in news articles.*

*A REST API, also called a RESTful API, is an API that follows REST principles. In a REST API, all data are treated as resources, each one represented by a unique uniform resource identifier (URI).* ***For example, the Twitter API makes each tweet an available resource that can be retrieved by clients. Clients can also use Twitter’s API to post tweets and perform other actions on the site.***

### ***Explain ‘Addressing’ in RESTful web services.***

### ***The process of locating various types of resources with the help of a URL on the REST server is known as ‘addressing’ in RESTful web services****. Usually, single or multiple resources are addressed by resources.*

### ***What are Idempotent methods?***

*Idempotent methods are known to return the same outcome even after the same request has been made multiple times, and* ***it avoids errors caused by duplicate requests on the client side****.*

### *What is the maximum payload size that can be sent in POST methods?*

*Theoretically, there is no such maximum limit for payload size that can be sent in POST methods. However,* ***payloads with larger sizes can consume larger bandwidth****. Thus the server could take more time to proceed with the request.*

### *Differentiate POST and PUT methods.*

*POST Method*

* *POST can create a resource on the server.*
* *POST is not idempotent.*
* *POST responses are cacheable.*

*PUT Method*

* *PUT is used to replace a resource at a specific URI with another resource.*
* *PUT is idempotent that it will only result in one resource even after calling it multiple times.*
* *PUT responses are not cacheable.*

### ***What are the most common HTTP response status codes you see while working in REST API?***

*Some of the most common response status codes are 200 OK, 201 Created, 400 Bad Request, 401 Unauthorized, 403 Forbidden, 404 Not Found, 500 Internal Server Error, 502 Bad Gateway, 503 Service Unavailable, etc.*

### ***What is caching in the REST API?***

*REST API stores a copy of a server response in a particular location of computer memory to retrieve the server response fast in the future. This method is temporary and called "caching."*

### ***What are “Options” in REST APIs?***

*It is an HTTP method used to fetch the supported HTTP options or operations that help clients to choose the options in REST APIs****. Cross-Origin Resource Sharing (CORS) uses the REST option method.***

### ***Different types of API architectures***

*There are other two API architectures used, SOAP (Simple Object Access Protocol), and RPC (Remote Procedure Call)*

### ***What does the HEAD method in REST APIs do?***

*The HEAD method is used to return the HTTP Header in read-only form and not the Body.*

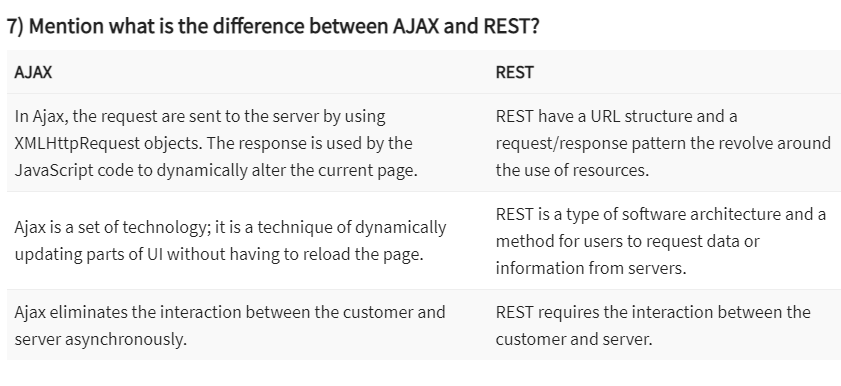
### ***Important aspects of RESTful web services implementation.***

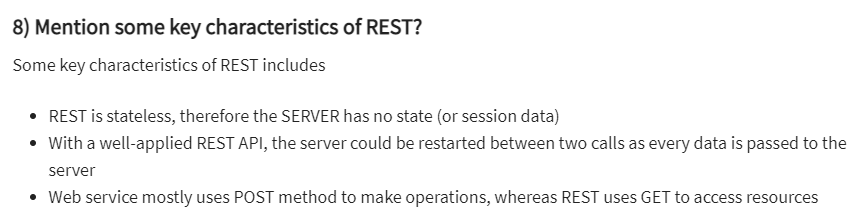
* *Resources Request*
* *Headers*
* *Request Body*
* *Response Body*
* *Status codes*

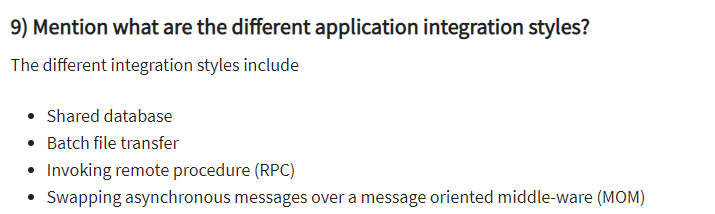
### ***Explain the architectural style for creating web API?***

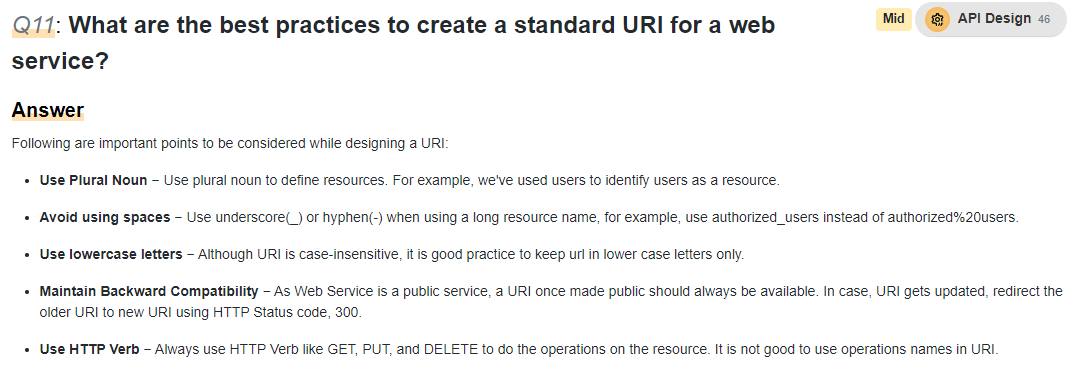
*The* ***architectural style*** *for creating web api are*

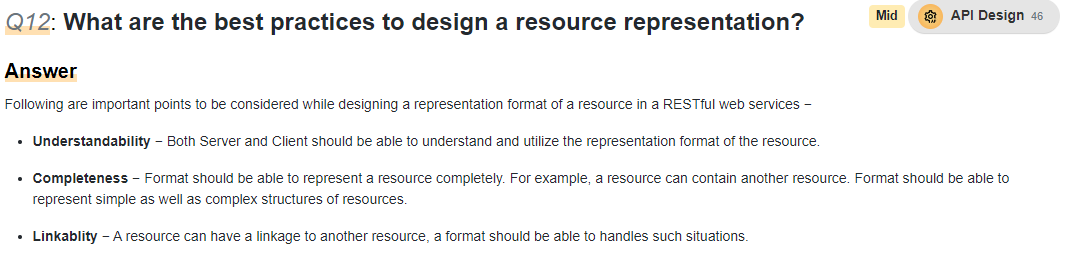
* *HTTP for client server communication*
* *XML/JSON as formatting language*
* *Simple URI as the address for the services*
* *Stateless communication*

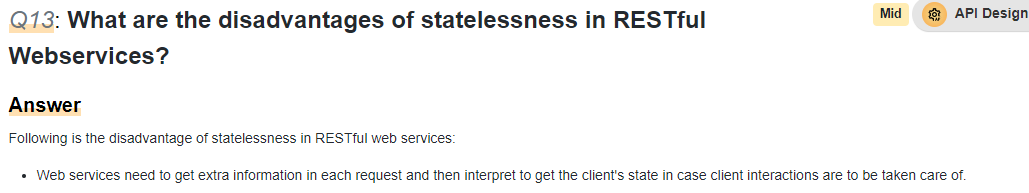
**

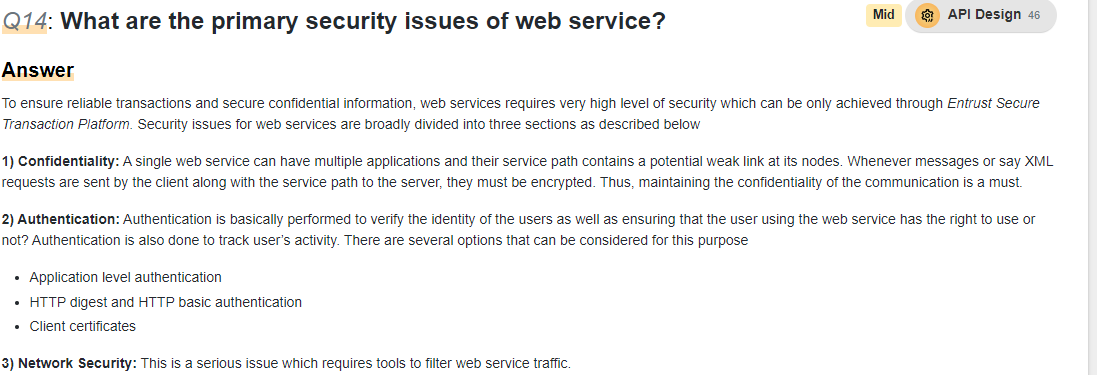
**

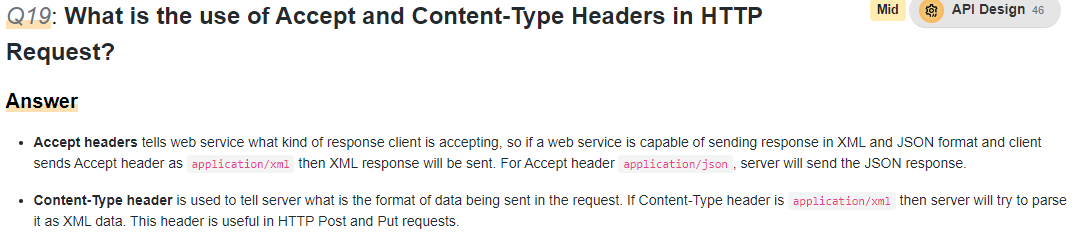
**

**

**

**

**

**

***Reference URL:***

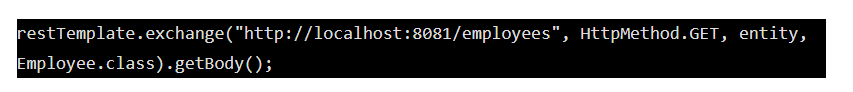
[***https://dzone.com/articles/top-20-spring-mvc-interview-questions-answers***](https://dzone.com/articles/top-20-spring-mvc-interview-questions-answers)

[*https://blog.hubspot.com/website/rest-api-interview-questions*](https://blog.hubspot.com/website/rest-api-interview-questions)

[*https://www.guru99.com/rest-api-interview-question-answers.html*](https://www.guru99.com/rest-api-interview-question-answers.html)

#### 14) What is RestTemplate and why do we use it?

*If you want to consume a RESTful service, then you can make use of RestTemplate. We use the exchange () method to invoke the service.*

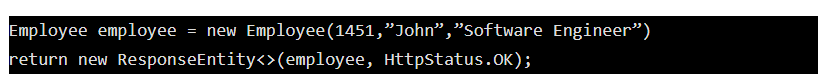
**

*First parameter to the exchange method is the URI, second parameter is the HTTP method i.e. GET, POST, PUT or DELETE, third parameter is the entity which consists of headers & body or just header/body and fourth parameter containing the return type of the response that the service will return.*

#### 15) What is ResponseEntity?

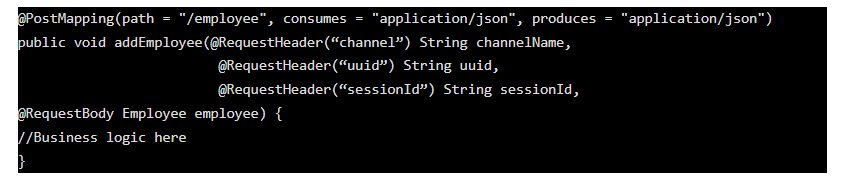
*Being a RESTful service provider if we want to return a response body that contains Data as well as the response code then we make use of ResponseEntity*

*E.g. let’s say we have a URI or a method* ***getEmployee*** *which returns an Employee object. Along with the Employee object we want to return a response code say 200, then we would return as*

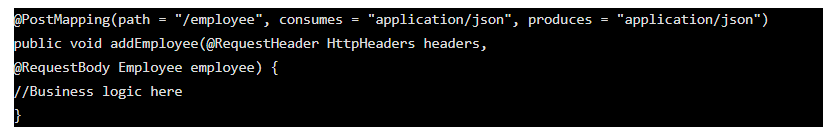
**

#### 16) @RequestHeader OR @RequestHeaders annotation

*Both the annotations are used to receive values from the request headers.-> @RequestHeader annotation is used to get each header one by one whatever is required.*

**

***@RequestHeaders****are used* ***to get all the headers into one object****. This can be used when there are too many headers and you want all instead of fetching it one by one.*

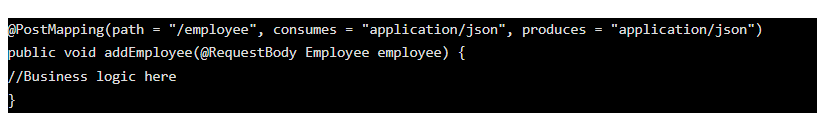
**

#### 17) What do you mean by Accept and Content-Type header

*Both are keys that are present in the headers sent from the client (E.g. browsers) to the service****Accept:*** *It contains the value which specifies what response format the client is expecting. E.g. application/json****Content-Type:*** *It contains the value which specifies which format of the data sent to the service.*

#### 18) Produces and consumes in REST

*Produces specifies the* ***MIME media type that is sent to the client*** *and Consumes specifies the MIME media type that is* ***accepted or consumed from the client****.  
E.g. The consumer of the below service has to send the request in the JSON format.*

**

#### 19) What are the best practises in developing RESTful services?

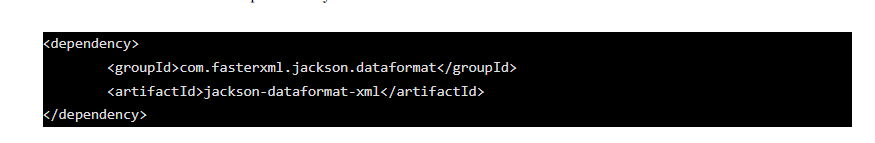
***As we know, there are two parties involved in REST communication****.* ***Service consumer and Service provider****. Service provider is the one who has exposed the web service and consumers are the one who are accessing the web service.****While developing we need to make the best use of HTTP methods, make semantic use of methods properly.****E.g. If you are providing data to the consumer then use HTTP GET, if you are creating a new resource then make use of HTTP POST.****Also make the best use of HTTP Response codes****. E.g. If the operation is successful send a 200 response. If the request format is not proper or some of the header is missing we need to send a 400 response. I did see some developers sending 500 Internal Server which is wrong. We need to send the correct response codes****Avoid sending sensitive data in the URIs and headers.******While designing the URIs, make use of plural forms. For example*** */emloyees/123 – CORRECT  
/employee/123 – WRONG*

#### 20) **What are HttpMessageConverters in Spring REST and why does it return JSON format by default?**

*HttpMessageConverters are used to convert the* ***object to JSON/XML and vice versa****. While developing REST service in Spring Boot, the default data format that is being returned in JSON.* ***This is because it registers the default MessageConvertor i.e. Jackson.***

#### 21) How to add XML support for our endpoint?

*As we know that JSON is the* ***default data format that REST service returns****, the question is how we send the* ***response in XML format*** *if the client requests it. If we are using spring boot then we need to add below dependency.*

**

#### 22) What is Swagger in the REST web services world?

*As per the* ***Open API/Swagger specification***“The **OpenAPI Specification** (OAS) defines a **standard, language-agnostic interface to RESTful APIs** which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection”***Swagger is a document*** *or* ***contract*** *which* ***is either .yml or .json format*** *which has below details  
-The list of endpoints present in our service and their respective HTTP method.  
-****Header*** *that we need to pass to the consumer.  
-****The data definition i.e. request body that a service consumer needs to send****, and in return what the endpoint/service provider will return.   
-****Http Response code*** *being returned for different scenarios.****Usually the service provider will prepare the document and share it with one or more service consumers,*** *so that both parties work independently based on the agreed contract.* ***Sometimes service providers & service consumers will discuss together to design the final contract/swagger file.***