**What does REST stand for?**

REST stands for the REpresentational State Transfer, which uses the HTTP protocol to send data from the client to the server, 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? (**[**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 [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. If you are new to Spring MVC, see this [Spring 5: Beginner to Guru](http://bit.ly/2ylC68g) resource to learn the basics.

**s 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 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.

**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 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?**([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 a request based upon HTTP methods, e.g. GET, POST, and various other parameters.

**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 the class annotated with the @Controller will also be automatically detected by the Spring container, as part of the container's component scanning process.

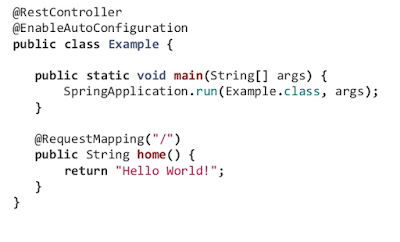
And, the @RestController  is a specialization of the @Controller for the RESTful web service. It not only combines the [@ResponseBody](http://courses.baeldung.com/p/rest-with-spring-the-master-class?affcode=22136_bkwjs9xa) and @Controller  annotations, but it also gives more meaning to your controller 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?**([answer](http://javarevisited.blogspot.sg/2017/08/difference-between-restcontroller-and-controller-annotations-spring-mvc-rest.html))

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.

[](http://bit.ly/2yjLysL)

**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 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:

1

@RequestMapping(path = "/hello", method = RequestMethod.PUT)

2

​

3

@ResponseBody

4

​

5

public String helloWorld() {

6

​

7

return "Hello World";

8

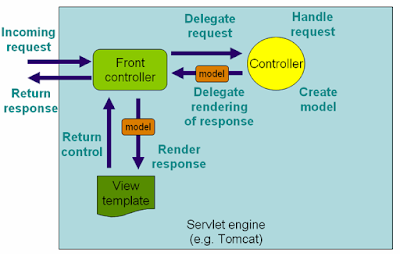
​

9

}

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.

**Spring MVC flow Diag:-**



**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))

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.

**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 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.

**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 an XML configuration.

It enables support for the @Controller-annotated classes that use @RequestMapping to map incoming requests to handler methods that are not already familiar with Spring's support for Java configuration.

**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 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)?**([answer](http://javarevisited.blogspot.sg/2012/01/rest-web-services-framework-interview.html))

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?**([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 by Java programmers with 1 to 2 years of experience in Spring. 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

Q: thin client vs REST client?

Q: java provides which utilities for rest client?

Q: what is rest api maturity model?

Q: @path?

Q: cache control httpheader to revalidate resource if max age passed?

Q: what secure token is used for service authentication?

Q: @queryparam?

Q: a restful webservice sends msg in form of “gopher request” and server responds back in form of a HTTP response?

Q: explain major codes 201,200,204,401, etc

Q: how to get HTTP request header in JAX-RS (2 ways)?

Q: @pathparam injects the value of URI paramaters that define in @path expression?

Q: \_\_\_\_\_\_\_ can be used to quickly check if a resource exists on the server or not?

Q: which HTTP methods is idemportant?