WEBSERVICES

JAX-RS is an specification (just a definition) and Jersey is a JAX-RS implementation. Imagine that there is a benevolent organization that creates (and gives away for free) designs for cars. Manufacturers around the world recognize that those designs are good and implement real cars based on those designs. Well, JAX-RS is just like a plan or design on how to build RESTful APIs, and software manufacturers recognize it and implement those ideas with real RESTful APIs, one of this is called Jersey, but there are others (i.e. RESTEasy). Just as many other manufacturers besides Toyota may use the designs to create their own version of cars.

Jackson is a Json Processor. It helps you in converting your objects to json and vice versa.

Web services are services that are exposed on the internet for programmatic access. They are online APIs that you can call from your code. RESTful web services are a type of web services that are modern, light-weight, and use a lot of the concepts behind HTTP, the technology that drives the web. they are web services, so the exchange of data happens over the web. Over HTTP. A client sends an HTTP request, and the server returns back an HTTP response. Similar to web sites. But like we discussed, instead of the response being complete web pages, only the data is returned, because the client is just a program, and not a human. The client could then have it’s own logic to present the data to the users in a presentable format, but the exchange between the web service client and web service server is usually just bare-bones data. When a web service client makes a request to a web service endpoint, they are usually messages transmitted from one machine to another. These messages need to be in a format, a language, that both the client and the server can understand. This is protocol. We saw that since RESTful web services are web services, the request and response messages are almost always exchanged over HTTP. But in HTTP, there are different methods available. You would have heard of GET, POST PUT and so on. Messages can be exchanged in any (or all) HTTP methods. There are guidelines and best practices that tells you what methods need to be used when designing the service, depending on what the request is, but there is no rule as such.

Representational state transfer is actually an architecture style. REST make good use of the ideas and concepts behind HTTP. HTTP stands for Hyper Text Transfer Protocol. Like we’ve already seen, you can think of a protocol as a language or mechanism for communication. So, HTTP is a way to exchange and communicate information online.

Rest have resource based URI.  A GET method lets you get information from the server. And POST is used when you want to submit information to the server. They work well with resource based URIs that we just saw. HTTP also defines status codes and response headers which lets the server send back extra information or metadata that might be useful to the client. HTTP also defines status codes and response headers which lets the server send back extra information or metadata that might be useful to the client. in the case of RESTful web services, you cannot send readable messages because the client is a piece of code! This is why sending the right status code is very important. ow can the server even identify what kind of data is sent? Similarly, how does the client know what data format is returned by the server? The answer is again a header value called Content-Type

This was a broad overview of some of the important points about RESTful web services and how they’ve been influenced by HTTP. When you design a RESTful API:

1. You need to have resource-based URIs. Every resource or entity should be identifiable by a single URI.
2. You need to choose the right HTTP methods for different actions and operations for the API.
3. The response needs to return the right HTTP status codes
4. All requests and responses need to have the right Content-Type header set so that the format of the messages are well understood by everyone.