WRITING EXERCISE #2 [Option 1]

Explaining Concepts

Understanding REST and SOAP Web Services

Web services emerged from several important trends over the past roughly 15-20 years, including distributed computing, the need for portable and interoperable applications that global businesses could deploy to meet business challenges, and the internet with its ubiquitous application infrastructure. While SOAP was much more prominent early on in the move toward web services, the REST model seems to have supplanted SOAP with broad service offerings for REST-based services from cloud providers, including Google and Amazon. Here is a brief technical overview of these two approaches.

SOAP	REST
Simple object access protocol	Representational state transfer
A protocol for remote method invocation that passes serialized objects	An architectural style that uses HTTP/S to invoke operations hosted as URIs
Legacy in remote procedure call (RPC), object request brokers (ORB), and other distributed-object computing mechanisms	Fundamental concept realized in the internet architecture itself, and its use of uniform resource identifiers (URIs).
Deployed to a web services container	Deployed to the internet infrastructure

Many systems running inside enterprise networks a decade or more ago were developed using the SOAP-based web services model. **SOAP** is an XML-based protocol format for remote method invocation over a network. That means that a client invoking the method (operation) remotely, over the network sends a SOAP message

As one example, the VMware vCenter management service included a Apache Axis SOAP implementation (running on vCenter's embedded Apache Tomcat application server engine) that exposed APIs to monitor virtual machines running under its control. The VMware vCenter SDK was available to customers using vCenter, and it included the basic elements needed for SOAP web services, such as the Web Services Description Language (WSDL), an XML file that described the web services, operations (methods) available, acceptable parameters for each operation, and their return types. Developers used the WSDL with Apache Axis client libraries and the Java JDK (or Microsoft .NET, for C# clients) to generate client-side proxy code ('stubs') from the WSDL for use by their client application code. The various artifacts would be compiled with the client application. At runtime, the client software would invoke operations against the remote server with the help of the proxy object, which formats the SOAP messages to send to the server and receives messages back from the server. In the SOAP model, actual objects are serialized by sender

and deserialized by recipient in the communication stream, which means that the object state is sent by client and server over the network in the SOAP model.

On the other hand, **REST** (representational state transfer) refers to an architectural style and does not define a protocol as such. It uses the foundation protocol of the internet, HTTP/S. This model was first described in 2000 by then-PhD candidate Roy Fielding, who took an overarching look at the internet as an application model unto itself. This model is built on the notion of resources that can be easily located by a uniform resource identifier (URI), such as their URL. In RESTful applications, APIs are represented as endpoints, essentially the URL that defines the API. Developers invoke RESTful APIs typically using the GET, POST, PUT, and other HTTP/S calls using some combination of HTML, Javascript, or JSON. Parameters are passed as name-value pairs along with the URL, such as "destinations=New+York+City" in this example from Google's Maps API:

```
https://maps.googleapis.com/maps/api/distancematrix/json?units=imperial&origins=Washington,DC&destinations=New+York+City,NY&key=YOUR_API_KEY
```

The service responds with the requested resource, which may include JSON, images, or web pages. (Using commercial APIs typically requires an API key, as shown in the string above). The end result is that the response, such as the specific map, is rendered in the context of the client browser.

REST May be Best for Fast and Wide Distribution

When developing a new web service, start by thinking about your intended audience and where and how you want to host your service. Especially if you plan to build a new web service around one of the many offerings of APIs and functionality offered by Google or Amazon, start by looking at their respective developer consoles and the API libraries already available. Rather than building a new web service from scratch, you may be able easily deploy a new web service using one of the many RESTful APIs available from the Google developer library. Especially if you want to take advantage of the browser-based metaphor for client applications or to provide new services, REST may be best.