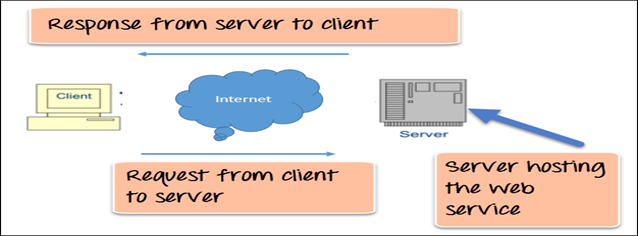
**Application Programming Interface (API)**

1. **What is an API?** 
   1. An application programming interface is an interface or communication protocol between different parts of a computer program intended to simplify the implementation and maintenance of software.
   2. Example: Object Oriented Design
      1. In object-oriented design, code is organized into objects. Your application my have hundreds of objects defined that can interact with one another. Each object has an API which is a set of public methods and properties that it uses to interact with other objects in your application. Now, an object might have inner logic that is set to private, which means that it is hidden from the outside scope, so this will not be an API.
2. **What is a RESTful API?**
   1. A RESTful API is an application program interface (API) that uses HTTP request to GET, PUT, POST, and DELTE data. A RESTful API will define a set of functions which developers can perform request and receive responses through HTTP protocols like GET and POST.
   2. Example: RESTful API
      1. Let’s say you are trying to find videos about Skydiving on YouTube. You will open YouTube, and type in skydiving into the search field, hit enter, and then you will see a list of videos about skydiving. A RESTful API will work in a similar way. You can search for something, and then you will get a list of results back from the service you are requesting.
3. **How does an API work?** 
   1. An API is a software negotiator that allows two applications to talk to each other. In other words, an API is the messenger that delivers your request to the provider that you are requesting if from and then delivers the response back to you.

**Introduction to Web Services**

1. **What is a Web Service?** 
   1. A Web Service is a service that is delivered over the web.
   2. Web Service is also a software system that is designed to be able to exchange and make use of information from machine-to-machine over a network.



1. **The 3 keys a Web Service Should Support are:** 
   1. Key 1: Designed for machine-to-machine (or application-to-application) interaction.
   2. Key 2: Should be platform independent.
   3. Key 3: Should allow communication over a network.
2. **How does data exchange between applications take place?** 
   1. The application will need to send an input called request to the web service, and then the web service will need to send an output response back to the application.

Input Request

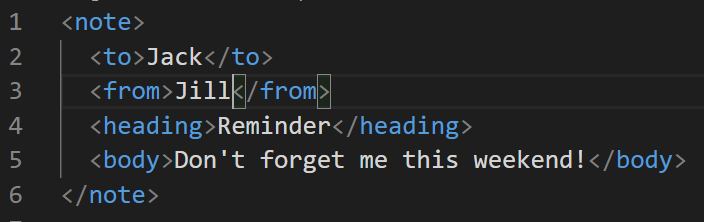
Application

Web Service

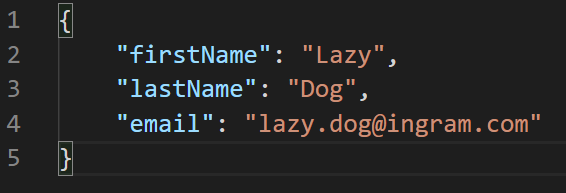
Output Response

* 1. In the image above the application will send a request “input” to the web service and then the web service will send a response “output” back to the application.

1. **What are the 2 popular formats for Request and Response?** 
   1. XML
      1. XML stands for extensible markup language. You can generate XML from Java, .net, or any other platform of your choice. XML is a software and hardware tool for storing and transporting data.
      2. Example: XML



* 1. JSON
     1. JSON stands for JavaScript Object Notation.
        1. JSON is text, and we can convert any JavaScript object into JSON, and then send that JSON to the server.
        2. JSON is supported by a wide variety of platforms. Not just front end but also backend as well.
     2. Example: JSON



1. **Key Terminology for Web Services.** 
   1. Request and Response
      1. Request is the input from the web service
      2. Response is the output from the web service
   2. Message Exchange Format
      1. The message format is the format for the request and the response. This is typically done in either XML or JSON.
   3. Service Provider
      1. The web service is called the service provider which is the one who holds the web service.
   4. Service Definition
      1. The service definition is the contract between the service provider and the service consumer.
   5. Transport
      1. Transport defines how a service is called which can be HTTP.

**Introduction to RESTful Web Services**

1. **What is a Restful Web Service?**
   1. REST is used to build Web services that are lightweight, maintainable, and scalable in nature. A service which is built on the REST architecture is called a RESTful service. The underlying protocol for REST is HTTP, which is the basic web protocol. REST stands for Representational State Transfer.
   2. REST is a way to access resources which are in an environment. For example, you could have a server that could be hosting important documents or pictures of videos. All of these are an example of resources. If a client, say a web browser need any of these resources, it must send a request to the server to access these resources. Now REST defines a way on how these resources can be accessed.
2. **The key elements of a RESTful implementation as follows:**
   1. **Resources:** These first key element is the resource itself. Let’s assume that a web application on a server has records of several employees. Let us assume the URL of the web application is <http://demo.hello.com>. Now, in order to access an employee record resource via REST, on can issue the command <http://demo.hello/1>. This command will tell the web server to please provide the details of the employee whose employee number is 1.
   2. **Request Verbs:** These describe what you want to do with the resource. A browser will issue a GET verb to instruct the endpoint it wants to get data. However, there are many other verbs available including things like POST, PUT, and DELETE. So, in the case of the example http://demo.com/employee/1, the web browser is issuing a GET Verb because it wants to get the details of the employee record.
   3. **Request Headers:** These are additional instructions sent with the request. These might define the type of response required or the authorization details.
   4. **Request Body:** Data is sent with the request. Data is normally sent in the request when a POST request is made to the REST web service. In a POST call, the client tells the web service that it wants to add a resource to the server. Hence, the request body would have the details of the resource which is required to be added to the server.
   5. **Response Body:** This is the main body of the response. So, in our example, if we were to query the web server via the request http://demo.hello.com/employee/1, the web server might return a JSON document with all the details of the employee in the Response Body.
   6. **Response Status Codes:** These codes are the general codes which are returned along with the response from the web server. An example is the code 200 which is normally returned if there is no error when returning a response to the client.
3. **RESTful Methods**
   1. Lest us assume that we have a RESTful web service that is defined at the location: <http://demo.hello.com/employee>. When the client makes any request to this web service, it can specify any of the normal HTTP verbs of GET, POST, DELETE, and PUT. Below is what would happen with the normal verbs that are sent by the client:
      1. **POST:** This would be used to create a new employee using the RESTful web service.
      2. **GET:** This would be used to get a list of all the employees using the RESTful web service.
      3. **PUT:** This would be used to update all the employees using the RESTful web service.
   2. Now, lest take a look at the RESTful web service that is defined with the location: <http://demo.hello/emoloyee/1>. This will be doing the same thing as the above statement, but it will be only accessing the first employee. So, if there were 2 employees you would access the second employee like <http://demo.hello/emoloyee/2>. You can also do the same thing as it pertains to the when the client makes any request to the web service, it can specify any of the normal HTTP verbs like GET, POST, DELETE, and put. Below is what would happen if we used the verbs GET, POST, DELETE, PUT on <http://demo.hello/employee/1>.
      1. **POST:** This would not be applicable since we are fetching data of employee 1 which is already created.
      2. **GET:** This would be used to get the details of the employee with employee number as 1 using the RESTful web service.
      3. **PUT:** This would be used to update the details of the employee with employee number as 1 using the RESTful web service.
      4. **DELETE:** This would be used to delete the details of the employee with the number as 1.
4. **Why RESTful Web Services?**
   1. Using RESTful Web services enables applications that are built on various programming languages to communicate with each other.

References:

<https://www.guru99.com/restful-web-services.html>