UNDERSTANDINGS & CONCLUSIONS

NANDINI BHATTACHARYA

MODULE – 6

27. Building Web API

27.1 Understanding HTTP Verbs

- HTTP verbs tell the server what to do with the data identified by the URL.
- The HTTP method is supplied in the request line and specifies the operation that the client has requested.
- In the context of an ASP.NET Web API resource, these four actions correspond to these verbs:
  1. **Request Verbs** -These describe what should be done with the resource. For more details on these verbs, you can refer this URL
  2. **Request Header** - When a client sends a request to the server, the request contains a header and a body. The request method contains additional information, such as - what type of response is required. For example: The response should be in XML, JSON, or some other format.
  3. **Request Body** - Request body contains the data that you want to send to the server. For example, a post request contains the data for a new item that you want to create. The data format may be in XML or in JSON.
  4. **Response Body** - The response body contains the data of the response from the server. For example, if the request is for a specific product, the response body includes product details in XML, JSON, and so on.
  5. **Response Status Codes** - These are the HTTP status codes that give the client details on the status of the request. Some of the common status codes are 404 not found, 204 No content, and so on.
- The **GET** method is designed to request a specific resource. In essence, it literally “gets” the resource in question, and is pretty limited to just that action. GET requests should only retrieve data, leaving other methods to perform the other transformative actions.
- **HEAD** is an interesting method in that it mirrors some functionality of another METHOD while unlocking additional possibilities. HEAD requests a GET response from a given resource, but with the response body excised. While that may seem overly simplistic, it allows greater flexibility and power for other API extensions – for instance, you can pass the headers of a resource to another request in an attempt to mimic a different requesting environment or situation, which is extremely helpful for testing and troubleshooting.
- **PUT** is somewhat the polar opposite of GET. While GET requests a specific resource, PUT places that resource in the remote directory. It should be noted that PUT assumes either the resource does not exist or the resource is fine to be overwritten – when using PUT, all representations of the target resource will be replaced by the payload.
- **PATCH** is designed to partially modify a targeted resource. In other words, while PUT places a resource in the target service, PATCH modifies that resource, as opposed to replacing it. This is a good way to update files or versions.
- **DELETE** deletes a targeted resource. The typical response for a deletion method is simply to reply with an “OK” status – either the resource was deleted, or it was not.
- **POST** allows an API to submit an attribute or entity to a given resource; in practice, this means the targeted resource receives a subordinate resource that is part of a larger collection.

27.3 Understanding JSON Structure

- JSON stands for JavaScript Object Notation
- JSON is a text format for storing and transporting data.
- e.g '{"Name":"Nandini", "Age":21}'
- You can receive pure text from a server and use it as a JavaScript object or send a JavaScript object to a server in pure text format.

- Simply, it is used for data-interchange. The benefit of JSON is that it has a very compact size as compared to XML documents of the same purpose and data.
- JSON stores the data in the form of key/value pairs.
- The keys are strings and the values are the JSON types. Keys and values are separated by colon. Each entry (key/value pair) is separated by comma.
- Another benefit of JSON is it is language-independent. You can work with JSON data in almost any programming language that can handle string objects.
- The basic structure of JSON document is very much simple. Every document in JSON must have either an object at its root, or an array.
- Object and array can be empty there is no need for it to contain anything in it, but there must be an object or an array.
- Array of objects: { "Name":"Nandini", "Age":21, "School":["KVS", "DPS", "St. Paul's"] }
- The **{** (curly brace) represents the JSON object. The **[** (square bracket) represents the JSON array. A JSON array can have values and objects.
- String typed values are the values which contain the character-type values. In many programming environments, strings are called, arrays of characters.
- In JavaScript, you can use either single quotes or double quotes to wrap the string values. But, in JSON specification you should always consider using double quotation.