**What is REST?**

It revolves around resource where every component is a resource and a resource is accessed by a common interface using HTTP standard methods. REST was first introduced by Roy Fielding in 2000.

In REST architecture, a REST Server simply provides access to resources and REST client accesses and presents the resources. Here each resource is identified by URIs/ global IDs.

**Mention some key characteristics of REST?**

Some key characteristics of REST (REpresentational State Transfer) includes

* REST is stateless, therefore the SERVER has no state (or session data)
* With a well applied REST API, the server could be restarted between two calls as every data is passed to the server
* Web service mostly uses POST method to make operations, whereas REST uses GET to access resources

**1) Explain what is REST and RESTFUL?**

REST uses various representations to represent a resource like text, JSON and XML. Now a days JSON is the most popular format being used in web services.

RESTFUL is referred for web services written by applying REST architectural concept are called RESTful services, it focuses on system resources and how state of resource should be transported over HTTP protocol to a different clients written in different language.

In RESTFUL web service http methods like GET, POST, PUT and DELETE can be used to perform CRUD operations.

**2) Explain the architectural style for creating web api?**

The architectural style for creating web api are

* HTTP for client server communication
* XML/JSON as formatting language
* Simple URI as the address for the services
* Stateless communication

**3) Mention what tools are required to test your web api?**

SOAPUI tool for SOAP WS and Firefox “poster” plugin for RESTFUL services.

**5) Mention whether you can use GET request instead of PUT to create a resource?**

No, you are not supposed to use POST or GET.  GET operations should only have view rights

**6) Mention what are resources in a REST architecture?**

Resources are identified by logical URLs; it is the key element of a RESTful design.  Unlike, SOAP web services in REST, you view the product data as a resource and this resource should contain all the required information.

**7) Mention what is the difference between AJAX and REST?**

|  |  |
| --- | --- |
| **AJAX** | **REST** |
| * In Ajax, the request are sent to the server by using XMLHttpRequest objects. The response is used by the JavaScript code to dynamically alter the current page * Ajax is a set of technology; it is a technique of dynamically updating parts of UI without having to reload the page * Ajax eliminates the interaction between the customer and server asynchronously | * REST have a URL structure and a request/response pattern the turn around the use of resources * REST is a type of software architecture and a method for users to request data or information from servers * REST requires the interaction between the customer and server |

**11) Mention which markup language can be used in restful web api?**

JSON and XML are the two markup language that can be used in restful web api

**15) Mention what is the difference between SOAP and REST?**

|  |  |
| --- | --- |
| **SOAP** | **REST** |
| * SOAP is a protocol through which two computer communicates by sharing XML document * SOAP permits only XML * SOAP based reads cannot be cached * SOAP is like custom desktop application, closely connected to the server * SOAP is slower than REST * It runs on HTTP but envelopes the message | * Rest is a service architecture and design for network-based software architectures * REST supports many different data formats * REST reads can be cached * A REST client is more like a browser; it knows how to standardized methods and an application has to fit inside it * REST is faster than SOAP * It uses the HTTP headers to hold meta information |

**Name some of the commonly used HTTP methods used in REST based architecture?**

Following well known HTTP methods are commonly used in REST based architecture −

* **GET** − Provides a read only access to a resource.
* **PUT** − Used to create a new resource.
* **DELETE** − Ued to remove a resource.
* **POST** − Used to update a existing resource or create a new resource.
* **OPTIONS** − Used to get the supported operations on a resource.

**What are webservices?**

A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner like inter-process communication on a single computer.

**What are RESTful webservices?**

Web services based on REST Architecture are known as RESTful web services. These web services use HTTP methods to implement the concept of REST architecture. A RESTful web service usually defines a URI, Uniform Resource Identifier a service, provides resource representation such as JSON and set of HTTP Methods.

**What is a Resource in REST?**

REST architecture treats every content as a resource. These resources can be text files, html pages, images, videos or dynamic business data. REST Server simply provides access to resources and REST client accesses and modifies the resources. Here each resource is identified by URIs/ global IDs.

**How to represent a resource in REST?**

REST uses various representations to represent a resource where text, JSON, XML. XML and JSON are the most popular representations of resources.

**What are the best practices to design a resource representation?**

Following are important points to be considered while designing a representation format of a resource in a RESTful web services −

* **Understandability** − Both Server and Client should be able to understand and utilize the representation format of the resource.
* **Completeness** − Format should be able to represent a resource completely. For example, a resource can contain another resource. Format should be able to represent simple as well as complex structures of resources.
* **Linkablity** − A resource can have a linkage to another resource, a format should be able to handles such situations.

**Which protocol is used by RESTful webservices?**

RESTful web services make use of HTTP protocol as a medium of communication between client and server.

**What is messaging in RESTful webservices?**

A client sends a message in form of a HTTP Request and server responds in form of a HTTP Response. This technique is termed as Messaging. These messages contain message data and metadata i.e. information about message itself.

**What are the core components of a HTTP Request?**

A HTTP Request has five major parts −

* **Verb** − Indicate HTTP methods such as GET, POST, DELETE, PUT etc.
* **URI** − Uniform Resource Identifier URI to identify the resource on server.
* **HTTP Version** − Indicate HTTP version, for example HTTP v1.1 .
* **Request Header** − Contains metadata for the HTTP Request message as key-value pairs. For example, client or browser or browser type, format supported by client, format of message body, cache settings etc.
* **Request Body** − Message content or Resource representation.

**What are the core components of a HTTP response?**

A HTTP Response has four major parts −

* **Status/Response Code** − Indicate Server status for the requested resource. For example 404 means resource not found and 200 means response is ok.
* **HTTP Version** − Indicate HTTP version, for example HTTP v1.1 .
* **Response Header** − Contains metadata for the HTTP Response message as key-value pairs. For example, content length, content type, response date, server type etc.
* **Response Body** − Response message content or Resource representation.

**What is URI?**

URI stands for Uniform Resource Identifier. Each resource in REST architecture is identified by its URI.

**What is purpose of a URI in REST based webservices?**

Purpose of an URI is to locate a resources on the server hosting the web service.

**What is format of a URI in REST architecture?**

A URI is of following format −

<protocol>://<service-name>/<ResourceType>/<ResourceID>

**What is the purpose of HTTP Verb in REST based webservices?**

VERB identifies the operation to be performed on the resource.

What are the best practices to create a standard URI for a web service?

Following are important points to be considered while designing a URI −

* **Use Plural Noun** − Use plural noun to define resources. For example, we've used users to identify users as a resource.
* **Avoid using spaces** − Use underscore\_\_ or hyphen−− when using a long resource name, for example, use authorized\_users instead of authorized%20users.
* **Use lowercase letters** − Although URI is case-insensitive, it is good practice to keep url in lower case letters only.
* **Maintain Backward Compatibility** − As Web Service is a public service, a URI once made public should always be available. In case, URI gets updated, redirect the older URI to new URI using HTTP Status code, 300.
* **Use HTTP Verb** − Always use HTTP Verb like GET, PUT, and DELETE to do the operations on the resource. It is not good to use operations names in URI.

**What is statelessness in RESTful Webservices?**

As per REST architecture, a RESTful web service should not keep a client state on server. This restriction is called statelessness. It is responsibility of the client to pass its context to server and then server can store this context to process client's further request. For example, session maintained by server is identified by session identifier passed by the client.

**What are the advantages of statelessness in RESTful Webservices?**

Following are the benefits of statelessness in RESTful web services −

* Web services can treat each method request independently.
* Web services need not to maintain client's previous interactions. It simplifies application design.
* As HTTP is itself a statelessness protocol, RESTful Web services work seamlessly with HTTP protocol.

**What are the disadvantages of statelessness in RESTful Webservices?**

Web services need to get extra information in each request and then interpret to get the client's state in case client interactions are to be taken care of.

**What do you mean by idempotent operation?**

Idempotent operation means their results will always same no matter how many times these operations are invoked.

**Which type of Webservices methods are to be idempotent?**

PUT and DELETE operations are idempotent.

**Which type of Webservices methods are to be read only?**

GET operations are read only and are safe.

**What is the difference between PUT and POST operations?**

PUT and POST operation are nearly same with the difference lying only in the result where PUT operation is idempotent and POST operation can cause different result.

**What should be the purpose of OPTIONS method of RESTful web services?**

It should list down the supported operations in a web service and should be read only.

**What should be the purpose of HEAD method of RESTful web services?**

It should return only HTTP Header, no Body and should be read only.

**What is caching?**

Caching refers to storing server response in client itself so that a client needs not to make server request for same resource again and again. A server response should have information about how a caching is to be done so that a client caches response for a period of time or never caches the server response.

**Which header of HTTP response, provides the date and time of the resource when it was created?**

Date header provides the date and time of the resource when it was created.

**Which header of HTTP response, provides the date and time of the resource when it was last modified?**

Last Modified header provides the date and time of the resource when it was last modified.

**Which header of HTTP response provides control over caching?**

Cache-Control is the primary header to control caching.

**Which header of HTTP response sets expiration date and time of caching?**

Expires header sets expiration date and time of caching.

**Which directive of Cache Control Header of HTTP response indicates that resource is cachable by any component?**

Public directive indicates that resource is cachable by any component.

**Which directive of Cache Control Header of HTTP response indicates that resource is cachable by only client and server, no intermediary can cache the resource?**

Private directive indicates that resource is cachable by only client and server, no intermediary can cache the resource.

**Which directive of Cache Control Header of HTTP response indicates that resource is not cachable?**

no-cache/no-store directive indicates that resource is not cachable.

**Which directive of Cache Control Header of HTTP response can set the time limit of caching?**

max-age directive indicates that the caching is valid up to max-age in seconds. After this, client has to make another request.

**What are the best practices for caching?**

Always keep static contents like images, css, JavaScript cacheable, with expiration date of 2 to 3 days. Never keep expiry date too high.

Dynamic contents should be cached for few hours only.

**What is the purpose of HTTP Status Code?**

HTTP Status code are standard codes and refers to predefined status of task done at server. For example, HTTP Status 404 states that requested resource is not present on server.

**What HTTP Status Code 200 states?**

It means, OK, shows success.

**What HTTP Status Code 201 states?**

It means, CREATED, when a resource is successful created using POST or PUT request. Return link to newly created resource using location header.

**What HTTP Status Code 204 states?**

It means, NO CONTENT, when response body is empty for example, a DELETE request.

**What HTTP Status Code 304 states?**

It means, NOT MODIFIED, used to reduce network bandwidth usage in case of conditional GET requests. Response body should be empty. Headers should have date, location etc.

**What HTTP Status Code 400 states?**

It means, BAD REQUEST, states that invalid input is provided e.g. validation error, missing data.

**What HTTP Status Code 401 states?**

It means, FORBIDDEN, states that user is not having access to method being used for example, delete access without admin rights.

**What HTTP Status Code 404 states?**

It means, NOT FOUND, states that method is not available.

**What HTTP Status Code 409 states?**

It means, CONFLICT, states conflict situation while executing the method for example, adding duplicate entry.

**What HTTP Status Code 500 states?**

It means, INTERNAL SERVER ERROR, states that server has thrown some exception while executing the method.

**WEB API**

**What is ASP.NET Web API?**

ASP.NET Web API is a framework that simplifies building HTTP services for broader range of clients (including browsers as well as mobile devices) on top of .NET Framework.

Using ASP.NET Web API, we can create non-SOAP based services like plain XML or JSON strings, etc. with many other advantages including:

* Create resource-oriented services using the full features of HTTP
* Exposing services to a variety of clients easily like browsers or mobile devices, etc.

**What are the Advantages of Using ASP.NET Web API?**

Using ASP.NET Web API has a number of advantages, but core of the advantages are:

* It works the HTTP way using standard HTTP verbs like GET, POST, PUT, DELETE, etc. for all CRUD operations
* Complete support for routing
* Response generated in JSON or XML format using MediaTypeFormatter
* It has the ability to be hosted in IIS as well as self-host outside of IIS
* Supports Model binding and Validation

**What New Features are Introduced in ASP.NET Web API 2.0?**

More new features introduced in ASP.NET Web API framework v2.0 are as follows:

* Attribute Routing
* External Authentication
* CORS (Cross-Origin Resource Sharing)
* OWIN (Open Web Interface for .NET) Self Hosting
* IHttpActionResult
* Web API OData

**WCF Vs ASP.NET Web API?**

Actually, **Windows Communication Foundation** is designed to exchange standard SOAP-based messages using variety of transport protocols like HTTP, TCP, NamedPipes or MSMQ, etc.

On the other hand, **ASP.NET API** is a framework for building non-SOAP based services over HTTP only.

**Is it True that ASP.NET Web API has Replaced WCF?**

It's a misconception that ASP.NET Web API has replaced WCF. It's another way of building non-SOAP based services, for example, plain XML or JSON string, etc.

Yes, it has some added advantages like utilizing full features of HTTP and reaching more clients such as mobile devices, etc.

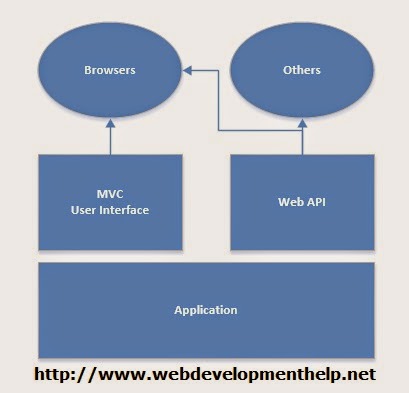
But WCF is still a good choice for following scenarios:

* If we intended to use transport other than HTTP, e.g. TCP, UDP or Named Pipes
* Message Queuing scenario using MSMQ
* One-way communication or Duplex communication

**MVC Vs ASP.NET Web API?**

As in previous ASP.NET Web API Interview Questions, we discussed that the purpose of Web API framework is to generate HTTP services that reach more clients by generating data in raw format, for example, plain XML or JSON string. So, ASP.NET Web API creates simple HTTP services that renders raw data.

On the other hand, ASP.NET MVC framework is used to develop web applications that generates Views as well as data. ASP.NET MVC facilitates in rendering HTML easy.

[](http://4.bp.blogspot.com/-kx7hZsksq5M/U2aC0cANOzI/AAAAAAAAKQc/5yNIt5QCqhs/s1600/MVCVsWebAPI.jpg)

**How to Return View from ASP.NET Web API Method?**

(A tricky Interview question) No, we can't return view from ASP.NET Web API method. We discussed in the earlier interview question about the difference between ASP.NET MVC and Web API that ASP.NET Web API creates HTTP services that renders raw data. Although, it's quite possible in ASP.NET MVC application.

**How to Restrict Access to Web API Method to Specific HTTP Verb?**

Attribute programming plays its role here. We can easily restrict access to an ASP.NET Web API method to be called using a specific HTTP method. For example, we may require in a scenario to restrict access to a Web API method through HTTP POST only as follows:

[HttpPost]

public void UpdateStudent(Student aStudent)

{

StudentRepository.AddStudent(aStudent);

}

**Can we use Web API with ASP.NET Web Form?**

Yes, ASP.NET Web API is bundled with ASP.NET MVC framework but still it can be used with ASP.NET Web Form.

It can be done in three simple steps as follows:

* Create a Web API Controller
* Add a routing table to Application\_Start method of Global.asax
* Make a jQuery AJAX Call to Web API method and get data

**How Can We Provide an Alias Name for ASP.NET Web API Action?**

We can provide an alias name for ASP.NET Web API action same as in case of ASP.NET MVC by using "ActionName" attribute as follows:

[HttpPost]

[ActionName("SaveStudentInfo")]

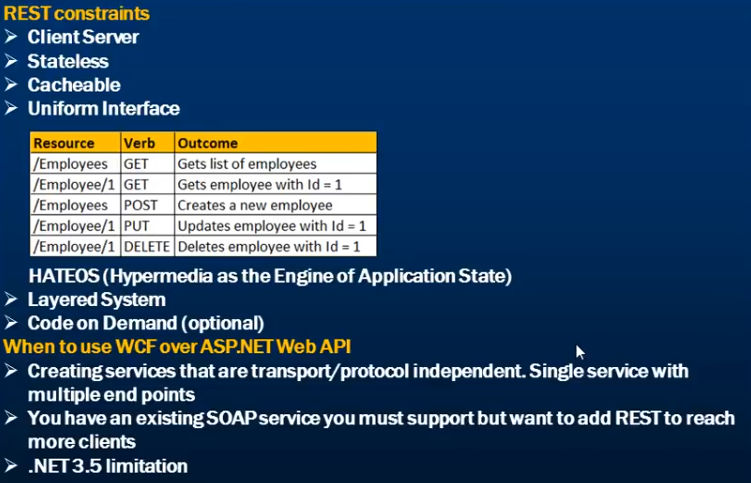
public void UpdateStudent(Student aStudent)

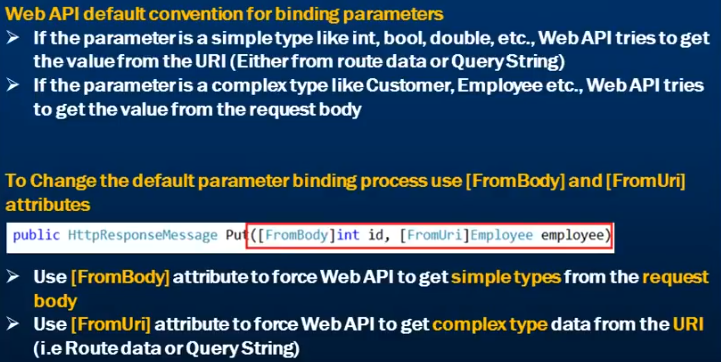
{

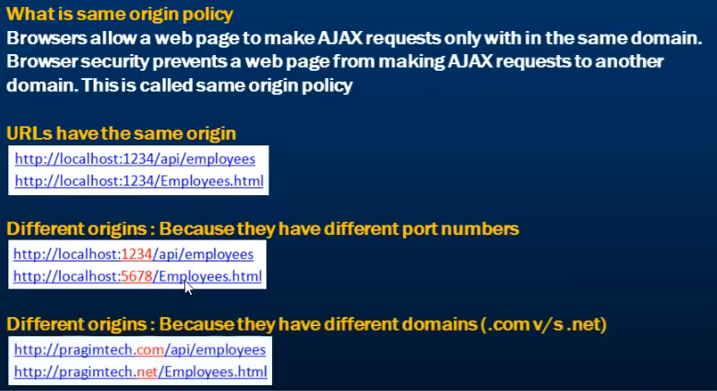
StudentRepository.AddStudent(aStudent);

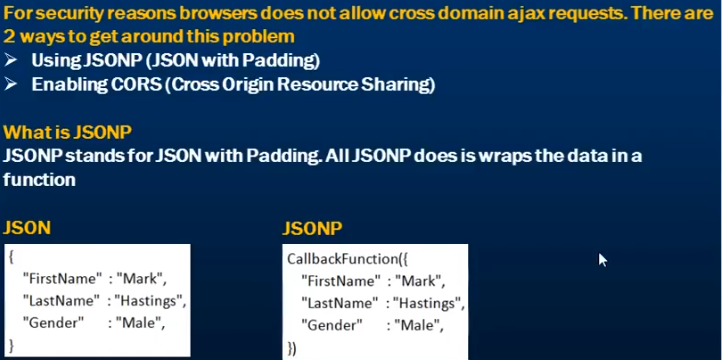
}

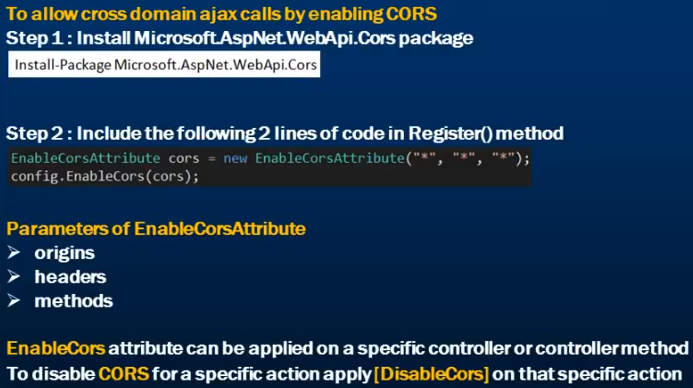




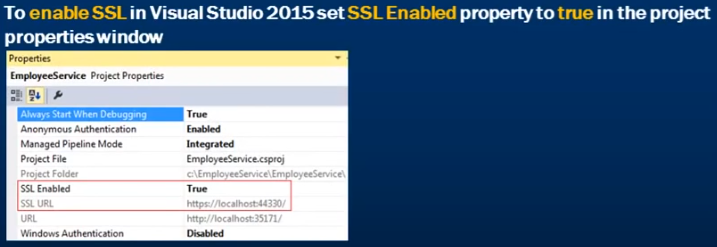
**[FromBody] & [FromUri]:**

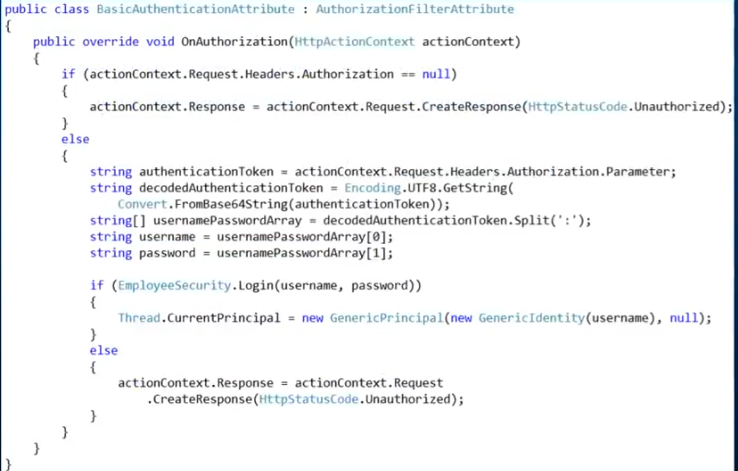
**Cross Domain:**

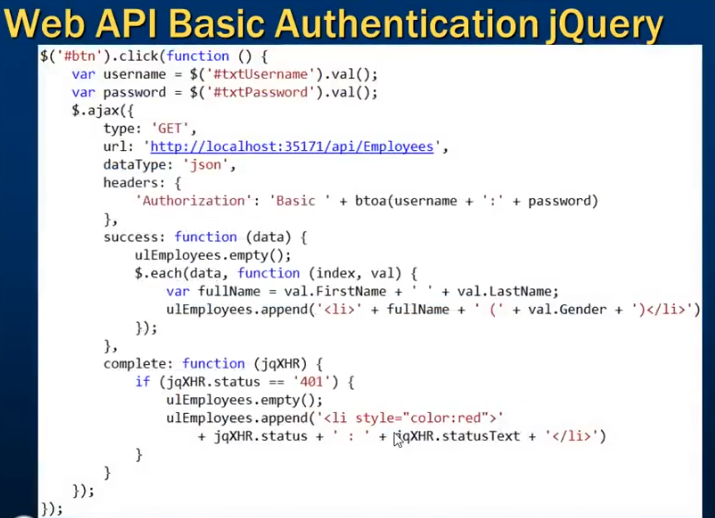


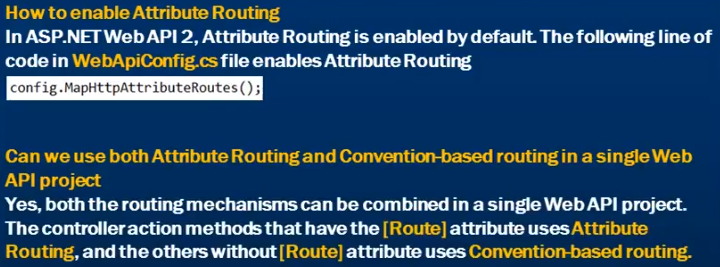
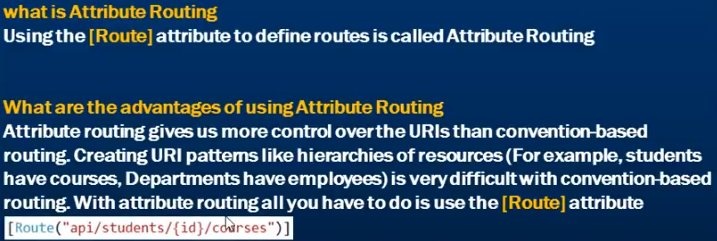
**Enabling CORS:**

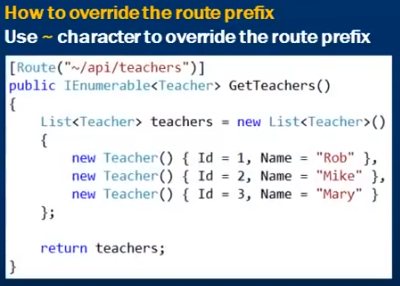
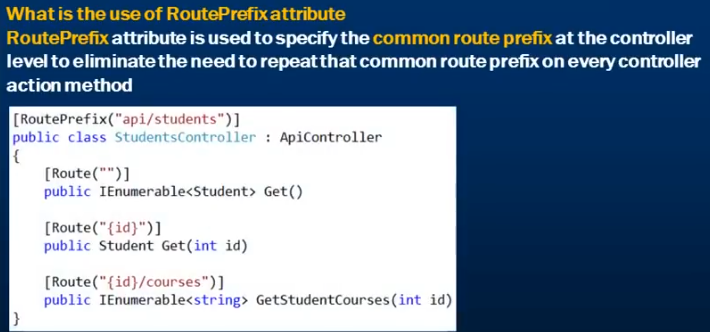
**Microsoft.AspNet.WebApi.Cors  
EnableCorsAttribute cors =new EnableCorsAttribute (origins, headers, methods)  
config.EnableCors(cors);**

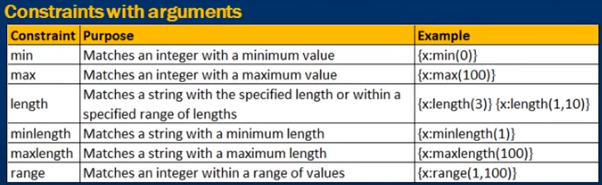
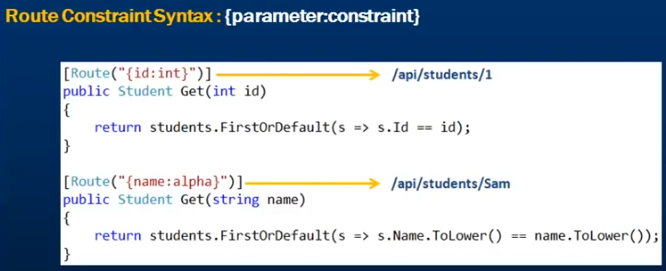


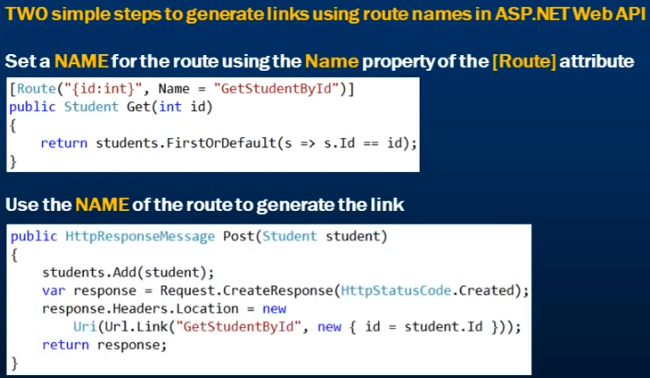
**Basic Authentication:**

btoa function encode the string into base 64

**Attribute Routing:**

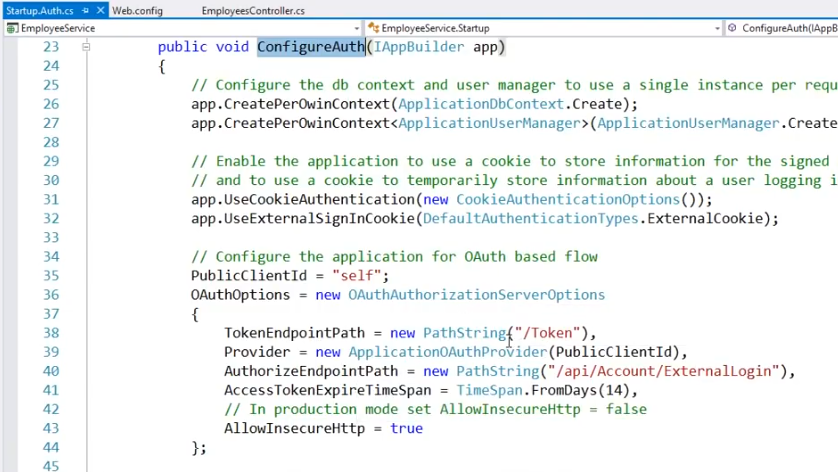
**Route Prefix:**

**Route Constraints:**



**Token Based Authentication:**Token-based authentication is a process where the user send his credential to the server, server will validate the user details and generate a token which is send as response to the users, and user store the token in client side, so client do further HTTP call using this token which can be added to the header and server validates the token and send a response.

Go to Startup.cs file under App\_Start folder in the solution



1. *// Configure the application for OAuth based flow*
2. PublicClientId = "self";
3. OAuthOptions = new OAuthAuthorizationServerOptions
4. {
5. TokenEndpointPath = new PathString("/Token"),
6. AuthorizeEndpointPath = new PathString("/api/Account/ExternalLogin"),
7. Provider = new ApplicationOAuthProvider(PublicClientId),
8. AccessTokenExpireTimeSpan = TimeSpan.FromDays(14),
9. *// In production mode set AllowInsecureHttp = false*
10. AllowInsecureHttp = false
11. };

Install the Owin using the below command in package manager console

1. Install-Package Owin -Version 1.0.0

**Owin :** Open web interface for .NET is a middleware which defines the interface between the web server and application.

1. **TokenEndPointPath :**This is a kind of request path, client applications communicate with server directly as part of the OAuth protocol.   
   It must begin with slash “/”
2. **Provider :**The object provided by the application to process the event raised by the authorization server middleware.
3. **AuthorizeEndpointPath :**The request path where the client application will redirect the client/user to obtain user account to issue a token
4. **AccessTokenExpireTimeSpan :**Defines the validity of token
5. **AllowInsecureHttp :** It will allow a normal http request to authorize, if it is set to false, it will process only https request.

**[Authorize] :** It is used to authenticate the token send from the client side, once the authentication is successfully the Get() will be fired

**authHeaders.Authorization = 'Bearer ' + resp.access\_token :**

| **Return type** | **How Web API creates the response** |
| --- | --- |
| void | Return empty 204 (No Content) |
| **HttpResponseMessage** | Convert directly to an HTTP response message. |
| **IHttpActionResult** | Call **ExecuteAsync** to create an **HttpResponseMessage**, then convert to an HTTP response message. |
| Other type | Write the serialized return value into the response body; return 200 (OK). |