WEB API  
The term API stands for ‘Application Programming Interface’. ASP.NET Web API is a framework for building Web API’s, i.e. HTTP based services on top of the .NET Framework. The most common use case for using Web API is for building RESTful services. These services can then be consumed by a broad range of clients like  
 1. Browsers   
2. Mobile applications   
3. Desktop applications   
4. IOTs

**REST** stands for Representational State Transfer.   
REST was first introduced in the year 2000 by Roy Fielding as part of his doctoral dissertation. REST is an architectural pattern for creating an API that uses HTTP as its underlying communication method. The REST architectural pattern specifies a set of constraints that a system should adhere to. Here are the REST constraints.  
 1. Client Server constraint   
2. Stateless constraint  
 3. Cacheable constraint   
4. Uniform Interface   
 Another concept related to Uniform Interface is HATEOAS. HATEOAS stands for Hypermedia as the Engine of Application State. All this means is that in each request there will be set of hyperlinks that let's you know what other actions can be performed on the resource.   
There are 2 other constraints   
1.Layered System  
2.Code on Demand (optional)

## **WCF vs Web API**

* Creating services that are transport/protocol independent -go WCF.
* You have existing SOAP service you must support but want to add Rest to reachc more clients- WCF
* You stuck to .Net 3.5 Limitation -go WCF

MediaTypeFormatter:   
 Base classs from which JSONMediaTypeFormatter, XMLMediaTypeFormater inherits from.

To return only XML:   


To return only JSON:



### **Implementing Get:**

If you want to specify a different names for you methods other the httpverbs(get,put, post, Delete)  
then you decorate your method with attributes.

Ex:

[HttpGet]



### **Implementing Post:**



### Implementing Put:



### **Query String Parameters:**

http://localhost:56968/api/employees/?gender=male



#### **FromBody, FromUri:**

By default Web API

for simple types: innt, char ,string, double, float .  
 it looks in query paramters [FromURI]

for complex types like customer, Employee,  
 it looks in RequestBody [FromBody]

ex: 

FromUri attribute tells web api to look inn query parameter  
FromBody attribute tells web api to in RequestBody

### **Implementing Delete:**



### **Ajax in Web API:**

Here we used relative URL (because both client and service lives in the same port/project)

If we want to make a api call from different project this won’t work because cross origin resource sharing is not configured.(cross domain ajax requests)



There are two ways to make CORS:

1. JSON P (Json with Padding): refer google ra
2. Enablinng CORS:  
   install cors nuget package and in webapiconfig.cs file add below two lines



This will enable cors for entire service for all origins(1st star) ,accepting all headers(2nd \*) for all methods(3rd star).

If you don’t want like that you can do like this



Or you can do like this



### **Authentication in Web API:**

There are 2 ways: basic authentication  
 Token authentication

Token authentication: decorate our API Get method with [Authorize] attribute

1. Register user:

![A screenshot of a social media post

Description generated with very high confidence]()![A screenshot of a social media post

Description generated with very high confidence]()

1. get token:

![A screenshot of a cell phone

Description generated with very high confidence]()

![A screenshot of a social media post

Description generated with very high confidence]()

Save this token in browser using jquery sessionStorage.setItem(‘user\_token’,response.access\_token);

Every time calling a request to GET Method use this token

![A screenshot of a cell phone

Description generated with very high confidence]()



Attribute Routing:

![A screenshot of a cell phone

Description generated with very high confidence]()



We have “api/students” string common in every route so we can extract that and specify RoutePrefix at the controller level ex: 

If we want separate Route under a routePrefix then see this:



RouteContraints:

![route constraints list.
Ref MSDN docs
]()



Generating Link(Response.headers.location)

![A screenshot of a social media post

Description generated with very high confidence]()



If we don’t add “/” at the end of the <http://localhost:56968/api/students> the we get location as <http://localhost:56968/api/students6> which is invalid. To solve this we need to set an name for the route and use that name to generate to link. Ex:



![A screenshot of a social media post

Description generated with very high confidence]()

IHttpActionResult vs HttpResponseMessage:

It(IHttpActionResult) is introduced in web API 2.0 which has helpers methods like ok(), Created(), which inherits from IhttpActionResult. To make unit testing easy and make code cleaner and easier

HttpResponseMessage exists in both 1.0 and 2.0 but IHttpActionResult only in 2.0

