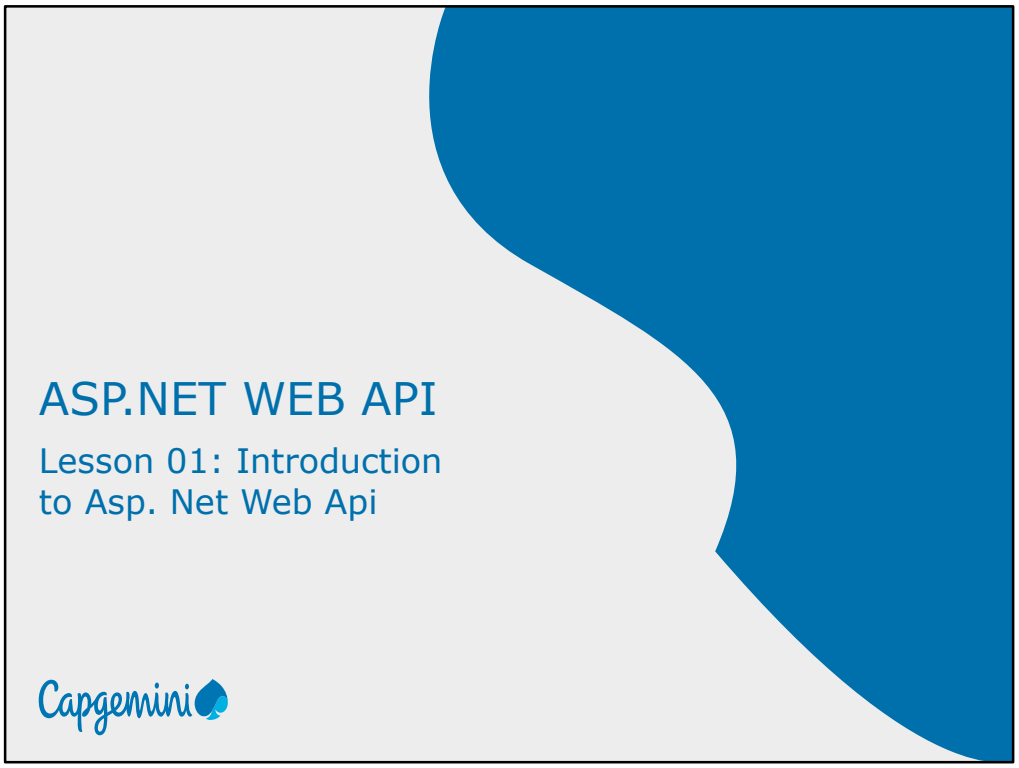

ASP.NET WEB API
Lesson 01: Introduction
to Asp. Net Web Api
Capgemini

Lesson Objectives

- In this lesson we will cover the following
  - Web API Features
  - SOAP- based Web Services
  - Web API Introduction
  - Web API Routing
  - Web API Parameter Biding
  - Content Negotiation

Overview

- The ASP.NET Web API makes it easy to create a set of web services that can respond to browser requests by using simple HTTP verbs such as GET, POST, and DELETE
- Using the Web API, you can build the back-end web services that a client-specific web application can call
- Building Web application by using client specific HTML Pages and the WEB API is an alternative to using ASP.NET MVC

ASP.NET WEB API

Asp. Net Web API is a framework for building HTTP services that can be consumed by a broad range of clients including browsers, mobiles, iphone and tablets. It is very similar to ASP.NET MVC since it contains the MVC features such as routing, controllers, action results, filter, model binders, IOC container or dependency injection. But it is not a part of the MVC Framework. It is a part of the core ASP.NET platform and can be used with MVC and other types of Web applications like Asp. Net Web Forms. It can also be used as an stand-alone Web services application.

X.1: Breadcrumb

SOAP-based Web Services

- SOAP Stands for " Simple Object Access Protocol "
- It is a specification for exchanging structured data i.e. XML using web services built on top of the HTTP Protocol
- As it relies on HTTP, it also follows the request and response model, where both request and response are represented by XML documents called messages

SOAP:-

SOAP Stands for Simple Object Access Protocol. It was proposed by Dave Winer and Don Box in 1988
SOAP was designed to solve the problem of hybrid platform enterprises where each system used a proprietary means of communication and was unable to connect to other systems. SOAP proposed the use of HTTP as the transport mechanism for sending and receiving XML.

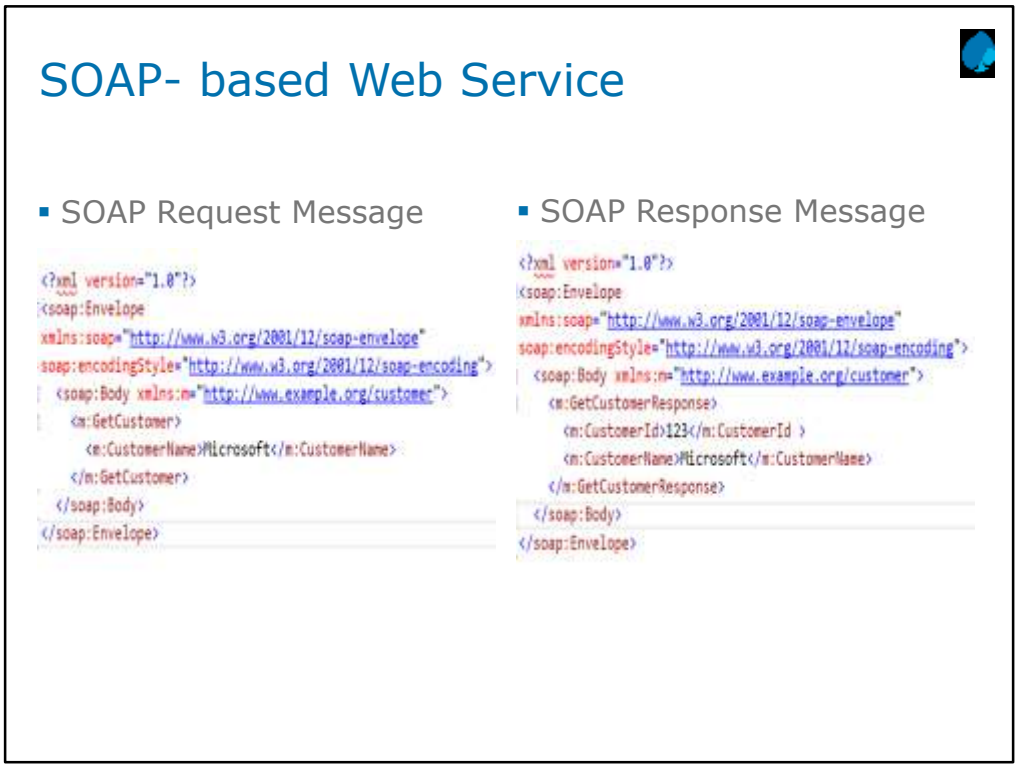
Following points briefly describes the nature of SOAP

•SOAP is a communication protocol designed to communicate via Internet.
•SOAP can extend HTTP for XML messaging.
•SOAP provides data transport for Web services.
•SOAP can exchange complete documents or call a remote procedure.
•SOAP can be used for broadcasting a message.
•SOAP is platform- and language-independent.
•SOAP is the XML way of defining what information is sent and how.
•SOAP enables client applications to easily connect to remote services and invoke remote methods.

Although SOAP can be used in a variety of messaging systems and can be delivered via a variety of transport protocols, the initial focus of SOAP is remote procedure calls transported via HTTP.

Other frameworks including CORBA, DCOM, and Java RMI provide similar functionality to SOAP, but SOAP messages are written entirely in XML and are therefore uniquely platform- and language-independent.

ASP.NET XML Web Services is an example of SOAP based Web Services which was introduced as a part of .NET Framework 2.0 and continued till .NET Framework 3.5.It uses SOAP for exchanging data between the client and server.



Above slide shows screenshot of Soap Request and Response Message

Soap Request Message represents a request to a customer service that returns a customer named **Microsoft**

Soap Response Message represents a response displaying CustomerID and CustomerName

A SOAP message is an ordinary XML document containing the following elements:

•An Envelope element that identifies the XML document as a SOAP message
•A Header element that contains header information
•A Body element that contains call and response information
•A Fault element containing errors and status information

X.1: Breadcrumb

HTTP Web Services

- HTTP is a flexible protocol that works very well on the internet where most us communicate.
- HTTP messages can be cached and travel through firewalls.
- They're lightweight which can be processed by processors on mobile devices.
- They can be encrypted and best of all nearly every programming environment in the world offers some capability to send and receive HTTP Messages.
- SOAP-based web services generally required a tool kit and more processing power where as HTTP is everywhere, it's lightweight and we can use it to exchange information with big servers as well as small device like mobile phones.

WCF is used to create Webservices using SOAP. WCF abstracts the way of underlying transport mechanism to exchange messages in the web service transaction.

In the Last few years implementing WEb services moving away from SOAP to webservices which embraces HTTP.

HTTP is a flexible protocol which can be cached, travel through firewall, light weight processed by all devices, it can be encrypted and nearly any programming environment in the world has some capabilities to send and recieve HTTP Messages where as SOAP based webservices requires a toolkit and more processing power but HTTP can be used anywhere.Due to this change in this Phase Microsoft concentrated for a new framework(part of ASP.NET) to support HTTP Webservices called Web API .

WCF can be used for writing Enterprise and can use tool kit where as WEB API is highly interoperable, extensible and light weight using HTTP.

X.1: Breadcrumb

Introduction to WEB API

- Web API is a framework that is part of ASP.NET MVC that enables you to build Representational State Transfer(REST) –enabled web services.
- REST enabled APIs help external systems use the business logic implemented in your application to increase the reusability of the application logic.
- Web API facilitates two way communication between client system and the server through task as
  - Instructing an application to perform a specific task
  - Reading data values
  - Updating data values

Web API is a framework that is part of ASP.NET MVC that enables you to build Representational State Transfer(REST) –enabled web services.

REST

REST stands for Representational State Transfer. This is a protocol for exchanging data over a distributed environment. The main idea behind REST is that we should treat our distributed services as a resource and we should be able to use simple HTTP protocols to perform various operations on that resource.

When we talk about the Database as a resource we usually talk in terms of CRUD operations. i.e. Create, Retrieve, Update and Delete. Now the philosophy of REST is that for a remote resource all these operations should be possible and they should be possible using simple HTTP protocols.

Now the basic CRUD operations are mapped to the HTTP protocols in the following manner:

GET: This maps to the R(Retrieve) part of the CRUD operation. This will be used to retrieve the required data (representation of data) from the remote resource.

PUT: This maps to the U(Update) part of the CRUD operation. This protocol will update the current representation of the data on the remote server.

POST: This maps to the C(Create) part of the CRUD operation. This will create a new entry for the current data that is being sent to the server.

DELETE: This maps to the D(Delete) part of the CRUD operation. This will delete the specified data from the remote server.

so if we take an hypothetical example of a remote resource that contain a database of list of books. The list of books can be retrieved using a URL like:

To retrieve any specific book, lets say we have some ID that we can used to retrieve the book, the possible URL might look like:

Since these are GET requests, data can only be retrieved from the server. To perform other operations, if we use the similar URI structure with PUT, POST or DELETE operation, we should be able to create, update and delete the resource form the server. We will see how this can be done in implementation part.

Now the if we compare the REST API wit SOAP, we can see the benefits of REST. Firstly only the data will be traveling to and fro from the server because the capabilities of the service are mapped to the URIs and protocols. Secondly, there is no need to have a proxy at the client end because its only data that is coming and the application can directly receive and process the data.

X.2: Breadcrumb

Introduction to WEB API Contd…

- Web API enable developers to obtain business information by using REST , without creating complicated XML request such as SOAP
- Web API uses url in requests and obtains results in the JSON format

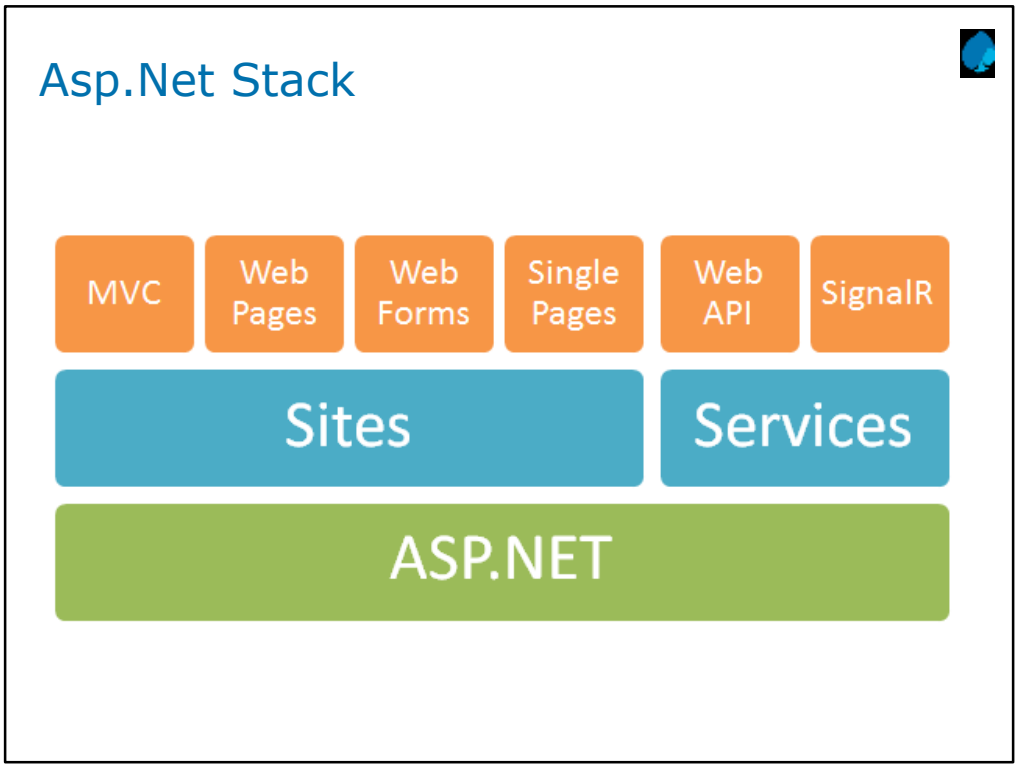
JSON:-

JSON stands for **J**ava**S**cript **O**bject **N**otation.
JSON is a syntax for storing and exchanging data.
JSON is an easier-to-use alternative to XML.
JSON is "self-describing" and easy to understand

Eg:-

```
[
{"Id":1,"Name":"Tomato soup","Category":"Groceries","Price":1.0},
{"Id":2,"Name“:"Yo-yo","Category":"Toys","Price":3.75},
{"Id":3,"Name":"Hammer","Category":"Hardware","Price":16.99}
]
```

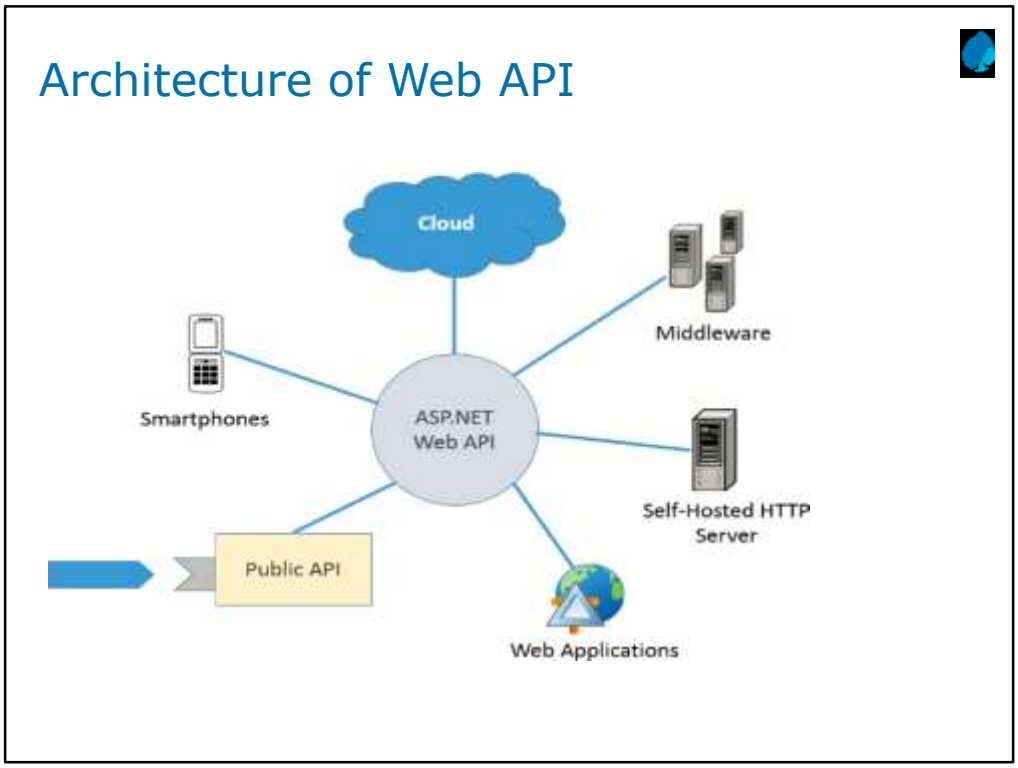
The above snippet show an example of JSON array having information about the products



The diagram above show the Asp.Net Stack which includes different type of application which part of ASP.Net .
The Asp.Net is divided in to two major type of applications :-

I) Sites which allows to create web application with user interface .It includes the following
   I) MVC :- allows to build application using MVC architecture
   II) Web Pages :- Web Pages is the simplest programming model for developing ASP.NET web pages. It provides an easy way to combine HTML, CSS, JavaScript and server code
   III) Web Form :- Web Forms is the oldest ASP.NET programming model, with event driven web pages written as a combination of HTML, server controls, and server code.
   IV) Single Pages :-Single Page Application (SPA) helps you build applications that include significant client-side interactions using HTML 5, CSS 3 and JavaScript.

I) Service which allows to create service to be used in web sites or web application.
   I) Web API :- ASP.NET Web API is a framework that makes it easy to build HTTP services that reach a broad range of clients, including browsers and mobile devices. ASP.NET Web API is an ideal platform for building RESTful applications on the .NET Framework.
   II) SingnalR :- SignalR can be used to add any sort of "real-time" web functionality to your ASP.NET application. ASP.NET SignalR is a library for ASP.NET developers that simplifies the process of adding real-time web functionality to applications. Real-time web functionality is the ability to have server code push content to connected clients instantly as it becomes available, rather than having the server wait for a client to request new data.



The above diagram show Architecture of Web API.

In the above diagram we can see that Asp.NET Web API allows to create services that can be consumed by a broad range of clients like browser, web application, smart phone , cloud , middleware and other public API's

Web Service v/s WCF v/s Web API

- Web Service
  - It is based on SOAP and return data in XML form.
  - It support only HTTP protocol.
  - It is not open source but can be consumed by any client that understands xml.
  - It can be hosted only on IIS.
- WCF
  - It is also based on SOAP and return data in XML form.
  - It is the evolution of the web service(ASMX) and support various protocols like TCP, HTTP, HTTPS, Named Pipes, MSMQ.
  - The main issue with WCF is, its tedious and extensive configuration.
  - It is not open source but can be consumed by any client that understands xml.
  - It can be hosted with in the applicaion or on IIS or using window service.

Web Service v/s WCF v/s Web API

- Web API
  - This is the new framework for building HTTP services with easy and simple way.
  - Web API is open source an ideal platform for building REST-ful services over the .NET Framework.
  - Unlike WCF Rest service, it use the full featues of HTTP (like URIs, request/response headers, caching, versioning, various content formats)
  - It also supports the MVC features such as routing, controllers, action results, filter, model binders, IOC container or dependency injection, unit testing that makes it more simple and robust.
  - It can be hosted with in the application or on IIS.
  - It is light weight architecture and good for devices which have limited bandwidth like smart phones.
  - Responses are formatted by Web API's MediaTypeFormatter into JSON, XML or whatever format you want to add as a MediaTypeFormatter.

X.X: [Topic]

Demo

- Demo :- Implementing a Simple WEB API Example

Add the notes here.

X.3: Breadcrumb

Consuming ASP.NET Web API

- Asp. Net API can be consumed from a variety of ways
- It can be consumed from any .NET Application using HttpClient class which is available in System.Net.Http Namespace
- To consumed an Web API from technologies like Java script and J query we have to use XMLHttpRequest

HttpClient Class:-

Provides a base class for sending HTTP requests and receiving HTTP responses from a resource identified by a URI.

XMLHttpRequest:-

XMLHttpRequest (**XHR**) is an API available to web browser scripting languages such as JavaScript. It is used to send HTTP or HTTPS requests to a web server and load the server response data back into the script.

X.X: [Topic]

Demo

- Demo :- Consuming an WEB API

Add the notes here.

X.3: Breadcrumb

Web API Routing

- Routing helps to map HTTP request to the Web API controllers and actions by using HTTP verbs and the request URL
- By default , routing rule in Web API is similarly to the routing rule in ASP.NET MVC
- We can make use of the naming convention to map request to actions or we can control the behavior of mapping by using annotations on action methods.
- Web API 2 support two types of Routing
  - Convention based Routing
  - Attribute Routing

Routing:-

In Asp.Net Web API Application, incoming HTTP request are handled by controller action methods. When an incoming request is received , it is routed to a proper action method and the routing system provides the fundamental layer for this activity.

The routing features in WEB API is one of the key component of the WEB API Framework it helps to locate controller and calls the action method of those controller by just using URLs.

To determine which action to invoke , the framework uses a routing table. This route is defined in the WebApiConfig.cs file which is placed at App_Start folder

The routing in WEB Api is similar to that of ASP.NET MVC

Convention Based Routing:-

•Convention based routing is the default routing rule for every Web API application .
•It is very convenient way of calling an Web API.
•It uses the default route defined for the API which is “api/{controller}/{id}”.
•Advantage of convention-based routing is that templates are defined in a single place, and the routing rules are applied consistently across all controllers.

X.3: Breadcrumb

Web API Routing

Attribute Routing :-

•Attribute routing is the new type of Routing which is introduced in Web API 2 .
•As the name implies attribute routing uses attribute to define routes.
•Attribute routing gives you more control over the URIs in your web API. For example, you can easily create URIs that describe hierarchies of resources.
•Unfortunately, convention-based routing makes it hard to support certain URI patterns that are common in RESTful APIs. For example, resources often contain child resources: Customers have orders, movies have actors, books have authors, and so forth.
•It's natural to create URIs that reflect these relations:
/customers/1/orders

This type of URI is difficult to create using convention-based routing. Although it can be done, the results don't scale well if you have many controllers or resource types.
With attribute routing, it's trivial to define a route for this URI.

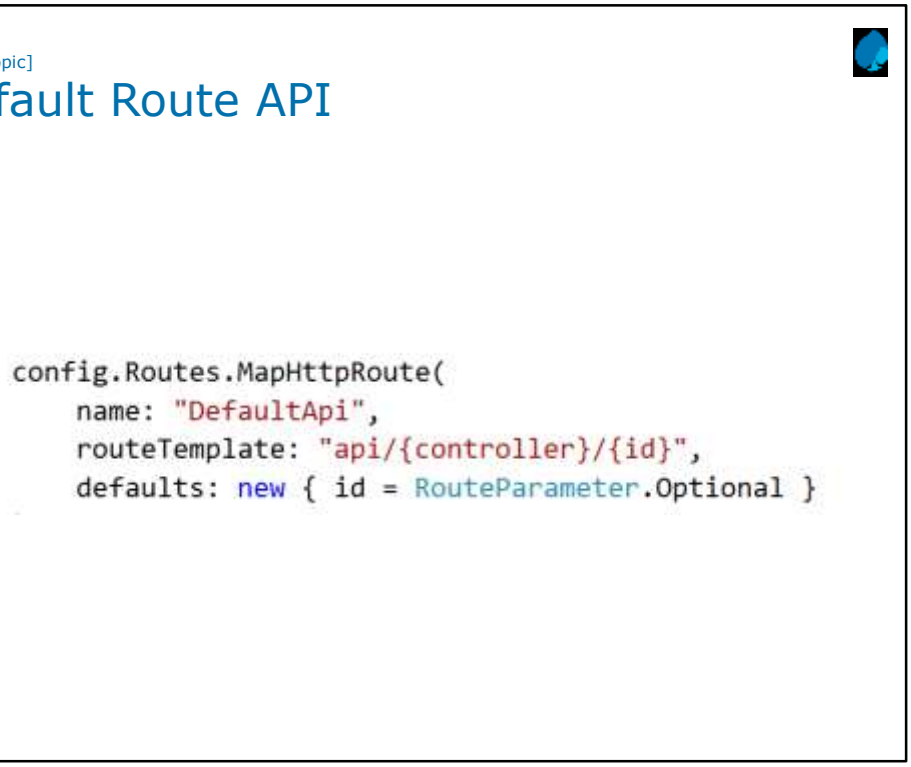
You simply add an attribute to the controller action:

```
[Route("customers/{customerId}/orders")]
public IEnumerable<Order> GetOrdersByCustomer(int customerId) { ... }
```

To enable attribute routing we use Route Attribute on the Action Methods as seen in the above example.

X.X: [Topic]

Default Route API



The above slide show code for the default route rule specified in WEB API project template.
In ASP.NET Web API, routes need to be defined on the **System.Web.Http.GlobalConfiguration.Configuration.Routes** property with the **MapHttpRoute** extension method, as shown in the previous section. Actually, there are other ways to register routes—the Add method of **HttpRouteCollection** class, for example. This method accepts two parameters: a string parameter for the name of the route and **IHttpRoute** implementation. Also, routes can be directly added into the **System.Web.Routing.RouteTable.Routes** static property with the **MapHttpRoute** extension method if the application is hosted under ASP.NET. We'll use the Routes static property on the GlobalConfiguration .
In the above example
The default route template for Web API is "api/{controller}/{id}". In this template, "api" is a literal path segment, and {controller} and {id} are placeholder variables. When the Web API framework receives an HTTP request, it tries to match the URI against one of the route templates in the routing table.
If no route matches, the client receives a 404 error.
For example, the following URIs match the default route:
/api/contacts
/api/contacts/1
/api/products/gizmo1
However, the following URI does not match, because it lacks the "api" segment:
/contacts/1
Note: The reason for using "api" in the route is to avoid collisions with ASP.NET MVC routing. That way, you can have "/contacts" go to an MVC controller, and "/api/contacts" go to a Web API controller. Of course, if you don't like this convention, you can change the default route table.

X.X: [Topic]

Demo

- Demo :- Implementing Routing in Asp,Net Web API

Add the notes here.

X.3: Breadcrumb

Web API Parameter Binding

- Parameter Binding provides a mechanism to get values from the URI and from the message body
- It allows to bind values to parameters when a controller are called
- The rules for binding the parameters depend upon the following types
  - Simple Types eg:- int ,bool, double,Datetime ,etc
  - Complex Type eg :- instance of a class

Parameter Binding :-

Parameter Binding allows the developer to read values to form the URI or from the message body when Web API Called. The parameter binding rule is dependent on the type of data to read.

These data can read directly form the URI or from the BODY of the Message

Simple Types :- If the parameter is the simple type then it is a string convertible parameter that includes the preemptive data types such as Int, Double, Bool and so on with the Date Time, Decimal, string and so on. By default these are read from the URI.

Complex Types :- If the parameter is a complex type then the Web API catches the value from the message body. It uses the media type formatters for catching the value from the body.

Consider the following code :-

```
HttpResponseMessage Get( int id ,Item item)
{

}
```

In the above example id is a Simple Type and item is a Complex Type.

Action Filters

- ActionName Attribute
  - Represents an attribute that is used for the name of an action.
- NoNAction Attribute
  - Represents an attribute that is used to indicate that a controller method is not an action method.

ActionName :-

ActionName action filter allow us to change name of action method while we are accessing it via url. By default we can access action method with {controller}/{action} pattern but in some case we don't want action method to be accessible by its name. So in such case we can use ActionName action filter attribute.

```
public class Items : ApiController
   {
     [HttpGet]
   [ActionName("Tulips")]
   public HttpResponseMessage GetTulipsImage(int Id);

   [HttpPost]
   [ActionName("Tulips")]
   public void AddTulipsImage(int id);
   }
```

NonAction:-

A non-action is an action that you want to prevent, in other words you do not want to call it. For that we use the NonAction attribute.

```
public class Items : ApiController
   {
[NonAction]
public string GetPrivateData() { }
}
```

X.3: Breadcrumb

Web API Parameter Binding Contd..

- Following attributes helps in Parameter Binding
  - FromUriAttribute
    - Specifies that an action parameter comes from URI of the incoming Http Request
  - FromBodyAttribute
    - Specifies that an action parameter comes only from the entity body of the incoming Http Request

Using [FromUri]
To force Web API to read a complex type from the URI, add the **[FromUri]** attribute to the parameter.
The following example defines a GeoPoint type, along with a controller method that gets the GeoPoint from the URI.

```
public class GeoPoint
{
public double Latitude { get; set; }
public double Longitude { get; set; }
}
public ValuesController : ApiController
{
public HttpResponseMessage Get([FromUri] GeoPoint location) { ... }
}
```

The client can put the Latitude and Longitude values in the query string and Web API will use them to construct a GeoPoint. For example:

```
http://localhost/api/values/?Latitude=47.678558&Longitude=-122.130989
```

Using [FromBody]
To force Web API to read a simple type from the request body, add the **[FromBody]** attribute to the parameter:

```
public HttpResponseMessage Post([FromBody] string name) { ... }
```

In this example, Web API will use a media-type formatter to read the value of *name* from the request body. Here is an example client request.

```
POST http://localhost:5076/api/values HTTP/1.1
User-Agent: Fiddler
Host: localhost:5076
Content-Type: application/json
Content-Length: 7
"Alice"
```

When a parameter has [FromBody], Web API uses the Content-Type header to select a formatter. In this example, the content type is "application/json" and the request body is a raw JSON string (not a JSON object).
At most one parameter is allowed to read from the message body. So this will not work:

```
// Caution: Will not work!
public HttpResponseMessage Post([FromBody] int id, [FromBody] string name) { ... }
```

X.X: [Topic]

Demo

- Demo:- Implementing Parameter Binding in Asp.NET Web API

Add the notes here.

X.X: [Topic]

Content Negotiation

- Content Negotiation is "the process of selecting the best representation for a given response when there are multiple representation is available"
- The primary mechanism for content negotiation in HTTP are these request headers which is as follows
  - Accept
  - Accept-Charset
  - Accept-Encoding
  - Accept-Language

Content Negotiation:-

"Content negotiation" is often used to describe the process of inspecting the structure of an incoming HTTP request to figure out the formats in which the client wishes to receive responses. Technically, though, content negotiation is the process in which client and server determine the best possible representation format to use in their interactions. Inspecting the request typically means looking into a couple of HTTP headers such as Accept and Content-Type. Content-Type, in particular, is used on the server for processing POST and PUT requests and on the client for choosing the formatter for HTTP responses. Content-Type is not used for GET requests.

The primary mechanism for content negotiation in HTTP are these request headers:
Accept: Which media types are acceptable for the response, such as "application/json," "application/xml," or a custom media type such as "application/vnd.example+xml"
Accept-Charset: Which character sets are acceptable, such as UTF-8 or ISO 8859-1.
Accept-Encoding: Which content encodings are acceptable, such as gzip.
Accept-Language: The preferred natural language, such as "en-us".

X.X: [Topic]

How Content Negotiation Works

- First, the pipeline gets the I content Negotiator service from the Http Configuration object. It also gets the list of media formatters from the Http Configuration. Formatters collection.
- Next, the pipeline calls I content Negotiatior .Negotiate, passing in:
  - The type of object to serialize
  - The collection of media formatters
  - The HTTP request
- The Negotiate method returns two pieces of information:
  - Which formatter to use
  - The media type for the response
- If no formatter is found, the Negotiate method returns null, and the client receives HTTP error 406 (Not Acceptable).

```
public class EmployeesController : ApiController
{
private static IList<Employee> list = new List<Employee>()
{
new Employee()
{
Id = 12345, FirstName = "John", LastName = "Human"
},
new Employee()
{
Id = 12346, FirstName = "Jane", LastName = "Public"
},
new Employee()
{
Id = 12347, FirstName = "Joseph", LastName = "Law"
}
};
// GET api/employees/12345
public Employee Get(int id)
{
return list.First(e => e.Id == id);
}
}
```

X.X: [Topic]

How Content Negotiation Works

Start Fiddler and go to the Composer tab. Issue a GET request for http://localhost:50568/api/employees/12345, specifying Accept: application/json in the Request Headers text box.

Request GET http://localhost:55778/api/employees/12345 HTTP/1.1
Accept: application/json
Host: localhost:55778
Response HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
{"Id":12345,"FirstName":"John","LastName":"Human"}

Issue a GET request for http://localhost:50568/api/employees/12345, specifying Accept: application/xml in the Request Headers text box. Now, the Web API response is XML.

Request GET http://localhost:55778/api/employees/12345 HTTP/1.1
Content-Type: application/xml
Host: localhost:55778
Response HTTP/1.1 200 OK
Content-Type: application/xml; charset=utf-8
<Employee xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://schemas.datacontract.org/2004/07/HelloWebApi.Models">
<FirstName>John</FirstName>
<Id>12345</Id>
<LastName>Human</LastName>
</Employee>
4. Issue a GET request for http://

X.X: [Topic]

Default Content Negotiator Class

- The components that govern the negotiation process in Web API is the called Default Content Negotiator Class
- The default implementation of I content Negotiator, which is used to select a Media Type Formatter for an Http Request Message or Http Response Message.
- I content Negotiator performs Content Negotiaition .

X.X: [Topic]

Demo

- Demo:- Implementing Content Negotiaition in Web API

Add the notes here.

Securing Web API

- Security is a very important concern for every application.
- In case of a Web Application it is very important and crucial concern for developer.
- Implementing Security is a very complex concept.
- In Asp.Net Web API security can be implement using Authentication and Authorization
  - Authentication helps to Authenticate a user against the application
  - Authorization help to check whether a Authenticated user is have rights or permission to perform specific action.

Security is always the top priority for web applications, because publicly accessible web applications are usually the targets of different types of web attacks. Hackers use web attacks to access sensitive information. To help prevent these attacks, you need to know how to use AntiXSS and request validation for your web application. Using various security mechanisms, you can build a secure web application. Another problem that developers usually face is the need to retain information amongst multiple postbacks. You need to know how to use state management techniques to retain information for multiple HTTP requests. Using state management, you can help reduce the need for users to re-enter information
every time they need to place a request.

Authentication :-

Authentication deals with a user's identity, verifying if the user, or connecting client, really is who he is claiming to be.

Authorization:-

Authorization deals with granting the user (connecting client) access to specific resources and system operations based on the user's identity.

Securing Web API

- Types of Authentication in Web API
  - Forms Authentication
  - Basic Authentication

Forms Authentication:-

Forms authentication uses the ASP.Net membership provider and uses standard HTTP cookies instead of the Authorization header. Forms authentication is not that REST-friendly as it uses cookies, and the clients would need to manage cookies to consume services that take advantage of forms authentication, which is vulnerable to cross-site forgery attacks. This is why you would need to implement CSRF measures if you use forms authentication. Forms authentication doesn't use encryption to secure the user's credentials. Hence, this is not a secure strategy unless you run your Web API over SSL.

Basic Authentication:-

Basic authentication sends the user's credentials in plaint text over the wire. If you were to use basic authentication, you should use your Web API over a Secure Socket Layer (SSL). When using basic authentication, we would pass the user's credentials or the authentication token in the header of the HTTP request. The service at the server side would need to parse the header to retrieve the authentication token. If the request is not a valid request, the server returns HTTP 401, meaning an unauthorized response.

Securing Web API

- Authorize Attribute
  - Authorize attribute is one of the of authentication filter which help us to implement authentication and authorization in Asp.Net Web API.
  - It is available in System.Web.Http namespace
  - Using this attribute we set an action of controller to be called only when a user has authenticated himself or herself.
  - We can also include user from a role or specific users to access the action method.

Securing Web API

- Example 1

```
public class EmployeesController:ApiController
{
   [Authorize]
    public HttpResposneMessage Post()
    {
      //Some Code
    }
  }
```

In the above we have decorate the Post method with the Authorize Attribute . When a user will call this method he will have authenticate himself first.

Securing Web API

- Example 2

```
public class EmployeesController:ApiController
{
   [Authorize(Users="Joydip,Jini")]
    public HttpResposneMessage Post()
    {
      //Some Code
    }
  }
```

The above code will restrict the method call to specific users

Securing Web API

- Example 3

```
public class EmployeesController:ApiController
{
   [Authorize(Roles="Administrator")]
    public HttpResposneMessage Post()
    {
      //Some Code
    }
  }
```

The above code will restrict the method call to users with Administrator role.

X.X: [Topic]

Lab

- Lab Topic

Add the notes here.

Summary

- In this lesson you have learnt about:
  - Web API Features
  - SOAP-based Web Services
  - HTTP Web Services
  - Web API Introduction
  - Web API Routing
  - Web API Parameter Binding
  - Content Negotiation

Add the notes here.

Review Question

- SOAP Stands for _____
  - Service Oriented Application Protocol
  - Simple Object Application Protocol
  - Simple Object Access Protocol
- Web API uses SOAP for Communication
  - True
  - False

Add the notes here.

Review Question

- Which of the following are the type of Binding in Web API?
  - Simple Binding
  - Convention Based Binding
  - Attribute Binding
  - Complex Binding

Add the notes here.