

# Session 07

## Design and Implementation of Web API

# Objectives

---

- Describe ASP.NET Web API
- Explain how to implement Web API
- Describe how to perform routing in Web API
- Explain how to implement database operations using Web API
- Describe dependency injection
- Explain how asynchronous and synchronous actions are implemented in Web API
- Describe OData services

# Overview of ASP.NET Web API

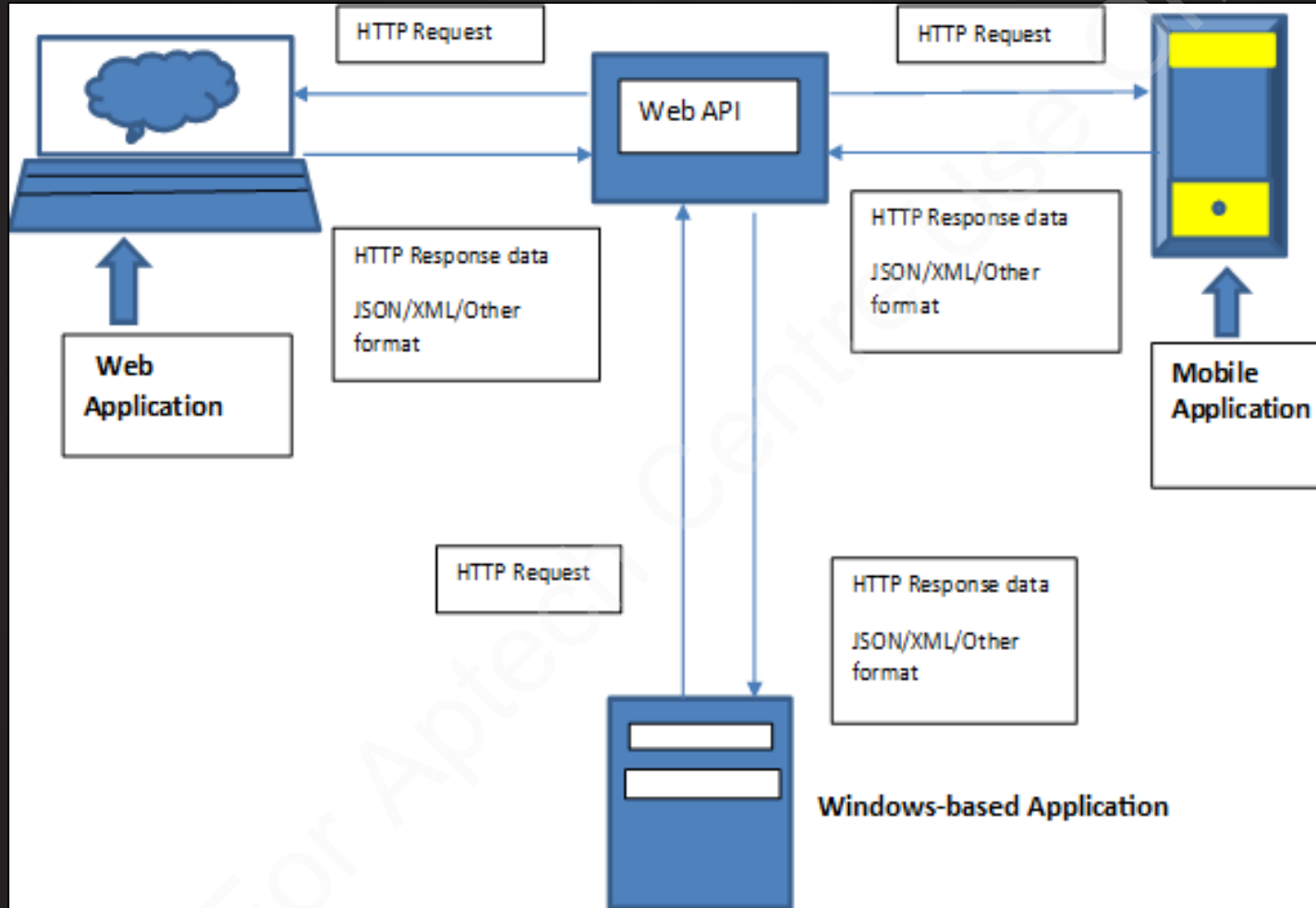
---

Refers to a framework that allows producing HTTP services easily

Is a technology provided by Microsoft that allows clients to create Web services targeting diverse clients

Client requests can be handled and responses can be sent back using content type, such as JavaScript Object Notation (JSON) or XML

# ASP.NET Web API [1-2]



# ASP.NET Web API [2-2]

---

## Key features of ASP.NET Web API:

### Simple Programming Model

Enables easy access and updates HTTP requests and responses in an ASP.NET Web API application.

### Content Negotiation

Enables client and server to negotiate data format that the service returns as a response.

### Request Routing

Provides a routing module that is responsible for mapping request URLs to the correct controller action.

### Filters

Provides filters to perform certain logic just before and after an action method of an API controller is invoked.

### Flexible Hosting

Provides flexibility to be hosted with other .NET applications, such as ASP.NET MVC and ASP.NET Web Forms.

# HTTP Request and Response [1-3]

Following are the four main HTTP methods:

Get	Post	Put	Delete
<ul style="list-style-type: none"><li>• Requests the server to retrieve a resource</li></ul>	<ul style="list-style-type: none"><li>• Requests the server that the target resource should process the data contained in the request</li></ul>	<ul style="list-style-type: none"><li>• Requests the server to create or update a request</li></ul>	<ul style="list-style-type: none"><li>• Requests the server to delete a resource</li></ul>

# HTTP Request and Response [2-3]

Following information is available in the HTTP request headers:

Host	<ul style="list-style-type: none"><li>• Specifies the domain name of the application whose resource is being accessed.</li></ul>
Accept	<ul style="list-style-type: none"><li>• Specifies the content type, also known as Multipurpose Internet Mail Extensions (MIME) type that the request expects as the response.</li></ul>
Accept Language	<ul style="list-style-type: none"><li>• Requests the server to create or update a request.</li></ul>
Connection	<ul style="list-style-type: none"><li>• Specifies whether the server should use the same connection (keep-alive value indicates this) for HTTP communication instead of creating a new one for each request.</li></ul>
Keep Alive	<ul style="list-style-type: none"><li>• Specifies the duration in seconds for which the server should use the same connection for HTTP communication.</li></ul>

# HTTP Request and Response [3-3]

Following are the key HTTP response headers:

Date	• Specifies the date and time when the response is being sent from the server.
Server	• Specifies the server that is handling the request response exchange.
Last modified	• Specifies the date and time at which the resource was last modified.
Content length	• Specifies the size of the response body, in decimal number of octets.
Content type	• Specifies the type of content that the response contains. In this example, the text/html value specifies the browser known to render the response body as HTML.



# Default Routing using Routing Tables

---

Visual Studio project has a template for Web API which creates a default route.



# Action Based Routing

---

A route can be created where the action name is included in the URI.

# Attribute Routing

---

- Attribute routing employs attributes for describing the definitions.
- Attribute routing provides flexibility for the URIs used in Web API.

For Aptech Centre Use Only

# Dependency Injection

---

Following are the three methods using which an object can be referenced to an external module:

## Constructor Injection

- In this case, a Class constructor has provision for dependencies.

## Setter Injection

- In this case, a setter method is used to inject the dependency by the client.

## Interface Injection

- In this case, the dependency is passed on to the client through an injector method. However, at the client's side, there must be an interface designed to accept a setter method of dependency.

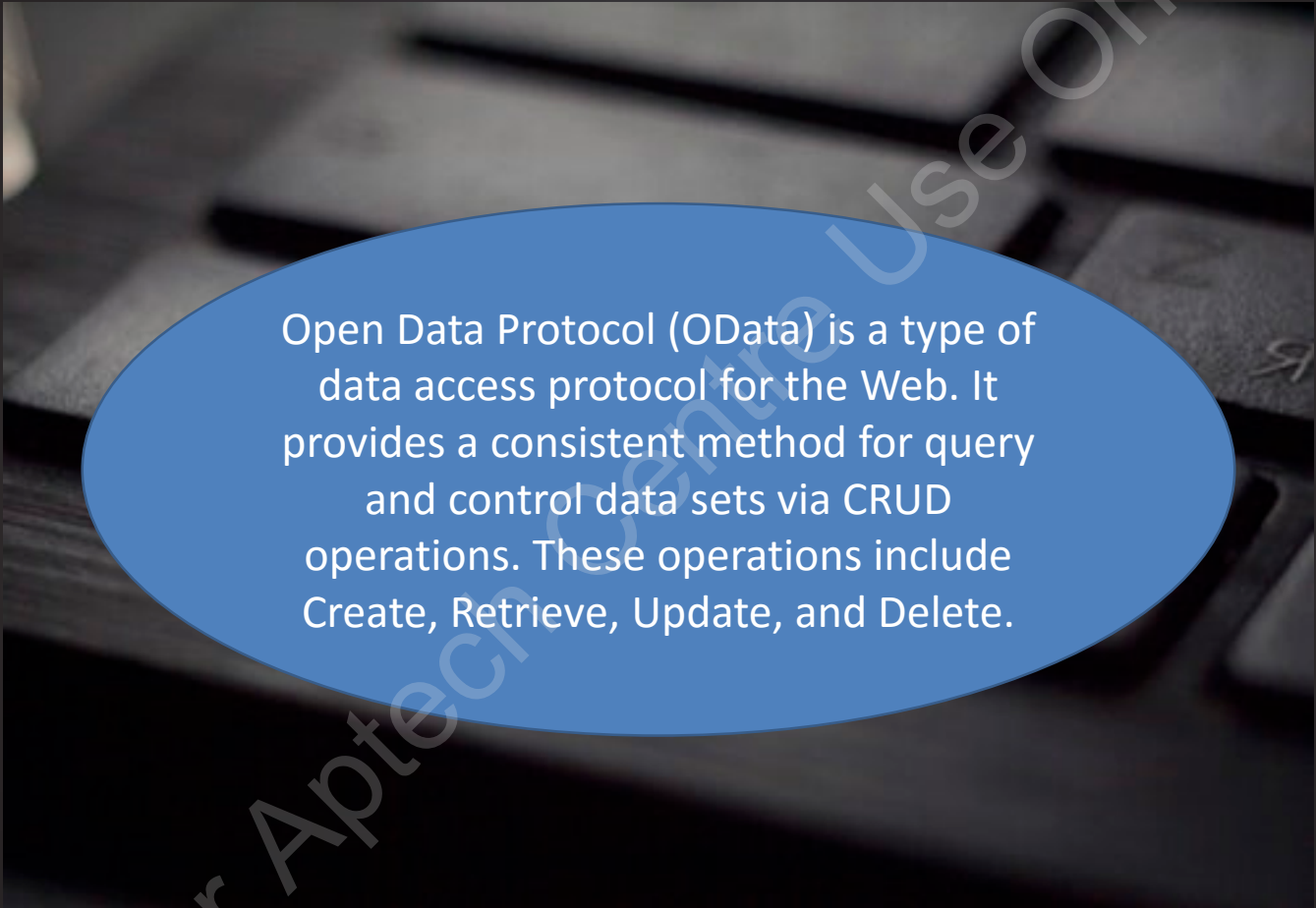
# Implementing Asynchronous and Synchronous Actions

---

- Asynchronous actions are a new addition in ASP.NET Framework.
- To implement them, Return type should be set to 'Task' or to Task<T>.
- As the method is asynchronous, the action also becomes asynchronous.
- The compiler executes the remaining methods in a continuous fashion or as a callback function.
- The callback function then executes another thread function from the Thread pool.

# OData Services

---



Open Data Protocol (OData) is a type of data access protocol for the Web. It provides a consistent method for query and control data sets via CRUD operations. These operations include Create, Retrieve, Update, and Delete.

# Summary

---

- ASP .NET Web API supports HTTP requests, such as GET, POST, PUT, and DELETE.
- The attributes of the Model View Controller (MVC), which include Routing, Controllers, Filters, Action results, Model binders, Inversion of Container (IOC), and Dependency injection are part of ASP.NET Web API.
- ASP.NET Web API supports OData functionality to implement easy CRUD functionality in databases.
- Routing in ASP .NET Web API works based on the routing table or action-based or attribute-based routing.
- ASP.NET Web API supports both synchronous and asynchronous methods of invocation.