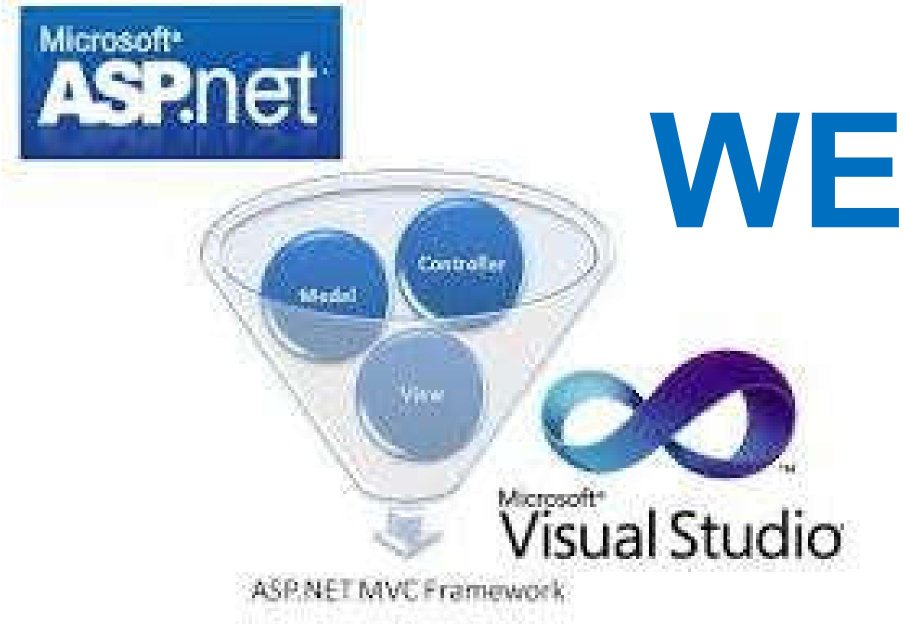
Section I



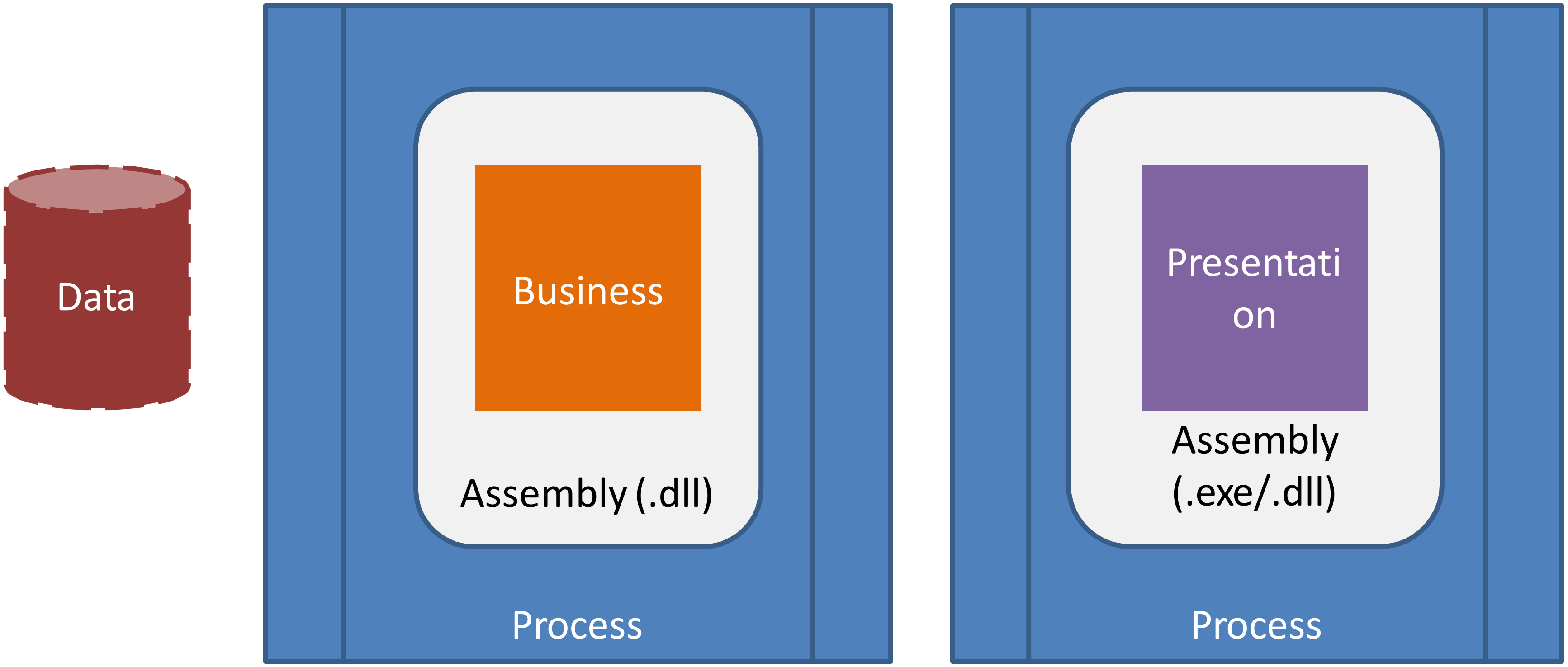
WEB-API

C#

1. ASP.NET WEB API INTRODUCTION

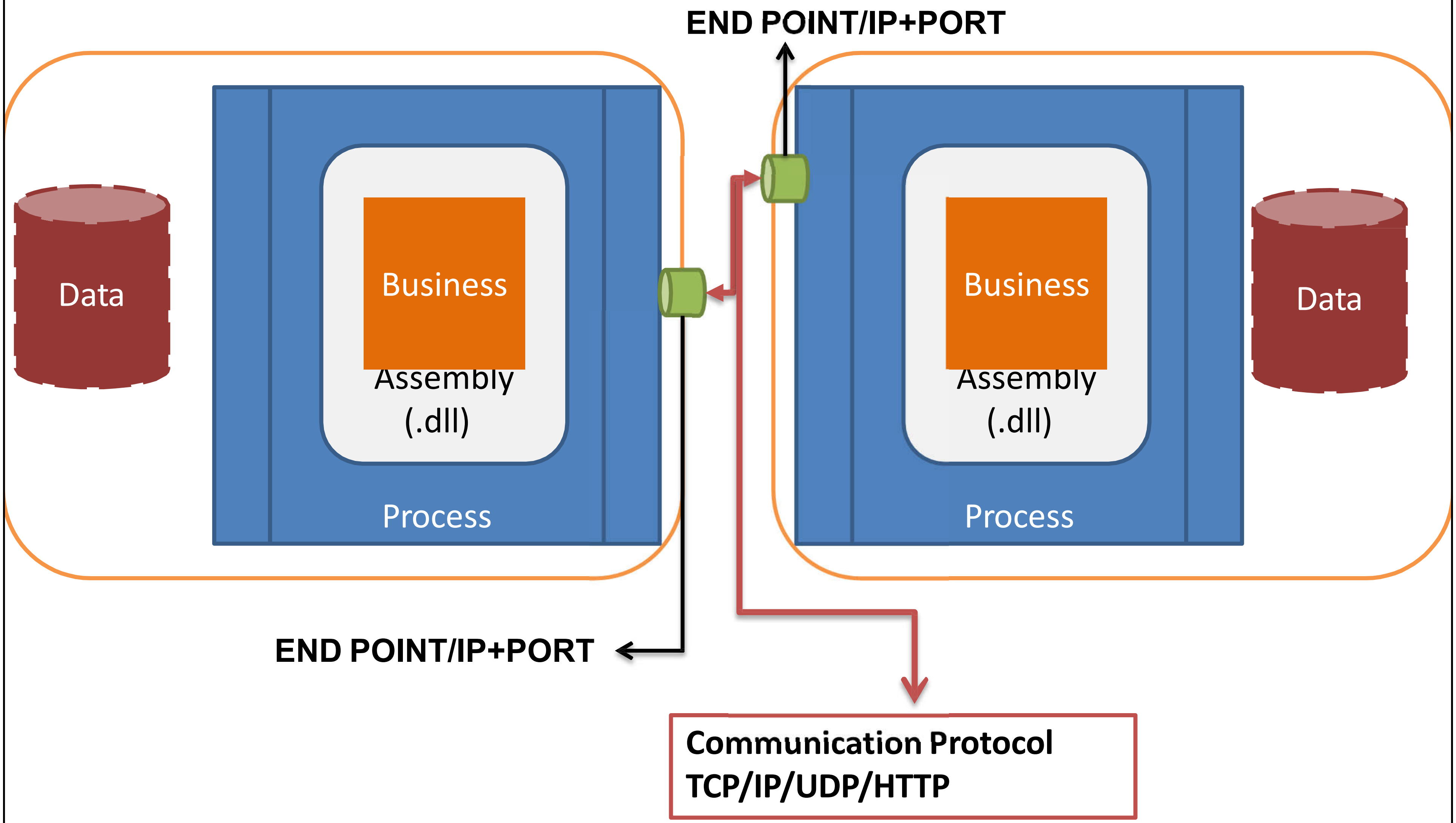
C#

Three Tier Architecture (Physical)



C#

InterProcess Communication



END POINT/IP+PORT

Data

Business Assembly (.dll)

Process

Business Assembly (.dll)

Process

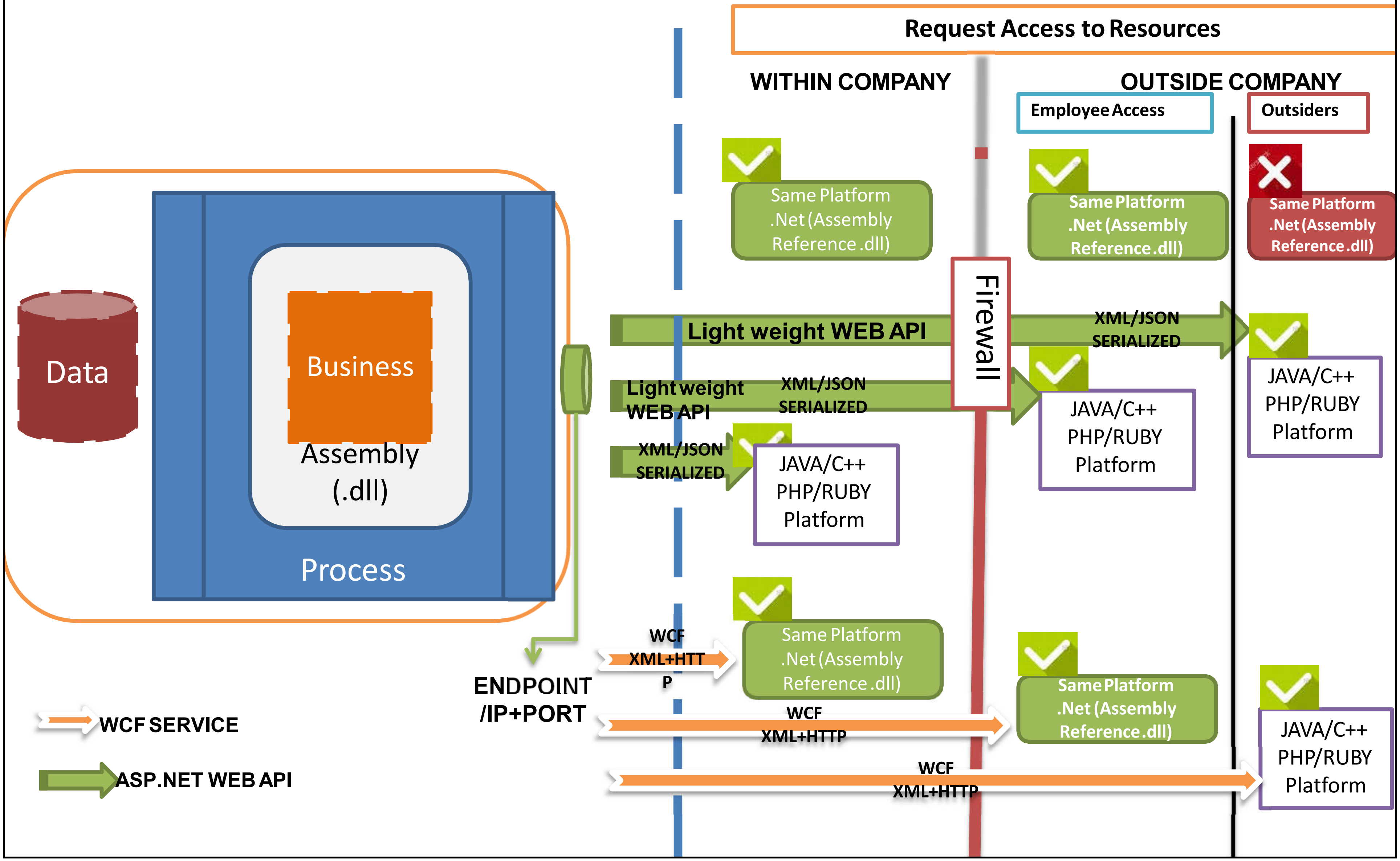
Data

END POINT/IP+PORT

Communication Protocol
TCP/IP/UDP/HTTP

C#

? NEED FOR WEB API/WCF



Request Access to Resources

WITHIN COMPANY

OUTSIDE COMPANY

Employee Access

Outsiders

Data

Business

Assembly (.dll)

Process

ENDPOINT /IP+PORT

Firewall

Same Platform .Net (Assembly Reference .dll)

Same Platform .Net (Assembly Reference .dll)

Same Platform .Net (Assembly Reference .dll)

Light weight WEB API

XML/JSON SERIALIZED

JAVA/C++ PHP/RUBY Platform

Light weight WEB API

XML/JSON SERIALIZED

JAVA/C++ PHP/RUBY Platform

XML/JSON SERIALIZED

JAVA/C++ PHP/RUBY Platform

WCF XML+HTTP

Same Platform .Net (Assembly Reference .dll)

WCF XML+HTTP

Same Platform .Net (Assembly Reference .dll)

WCF XML+HTTP

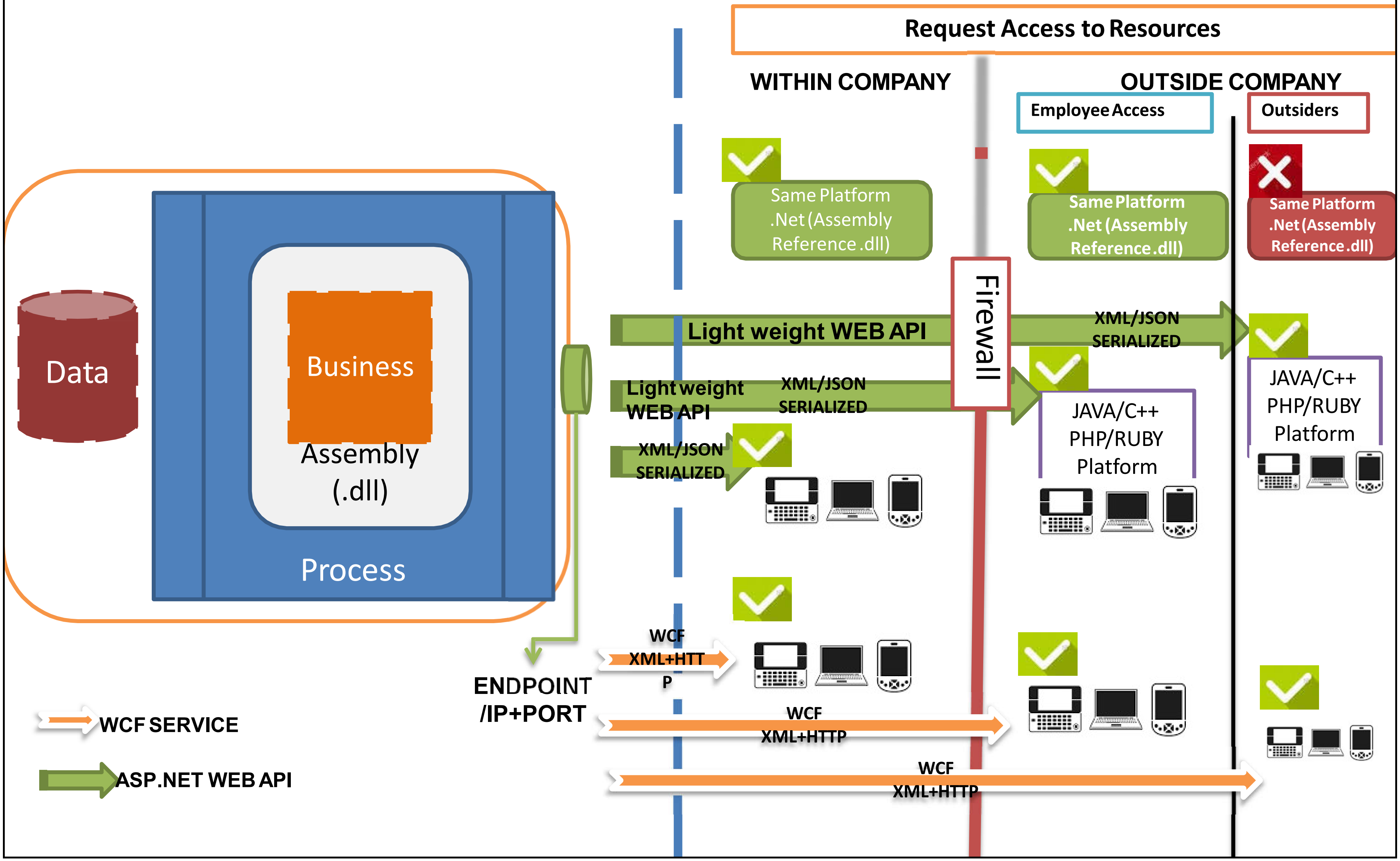
JAVA/C++ PHP/RUBY Platform

WCF SERVICE

ASP.NET WEB API

C#

? NEED FOR WEB API/WCF



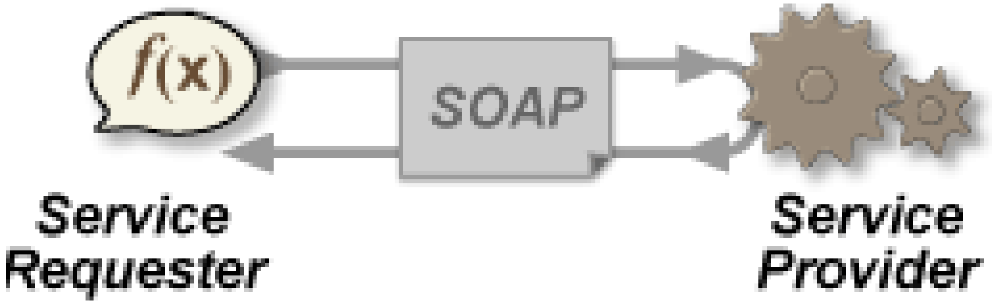
ASP.NET WEB API

Web API is the great framework for exposing your data and service to different-different devices. it use the full features of HTTP (like URIs, request/response headers, caching, versioning, various content formats)

Web services

- A web service is a collection of protocols and standards used for exchanging data between applications or systems.

WebServices are published, described and located over Internet.

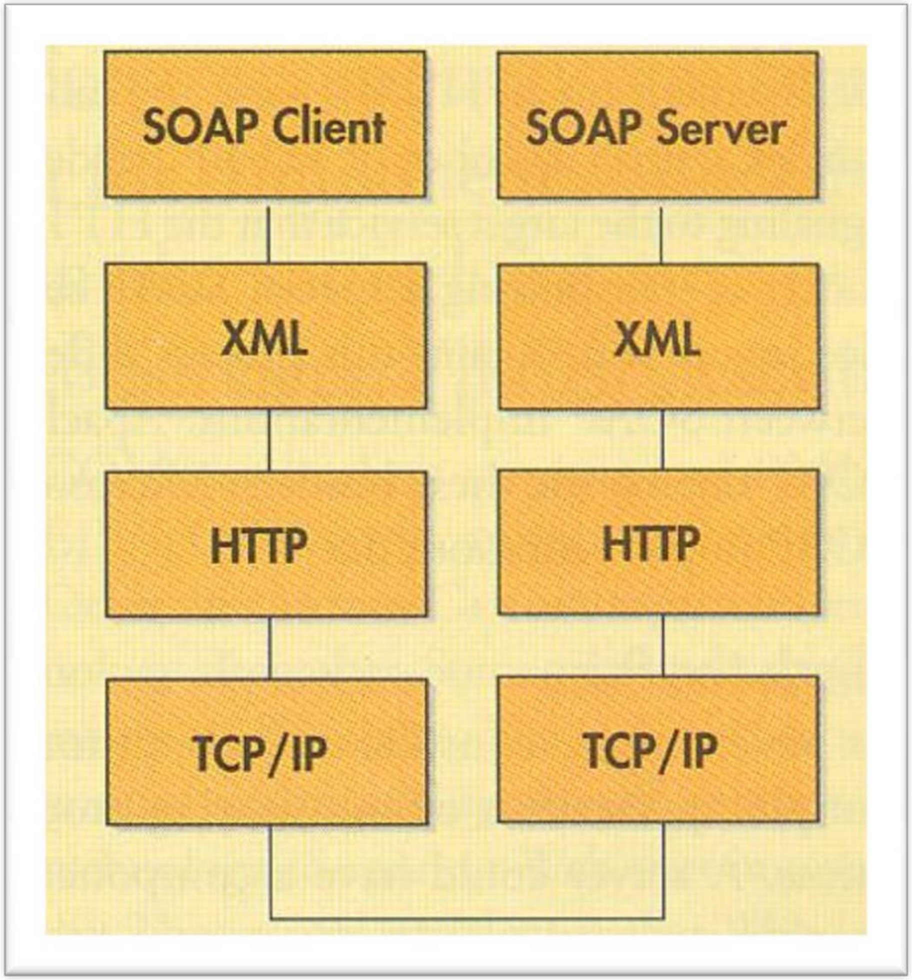


Characteristics

1. A Web Service is **accessible over the Web**.
2. Web Services communicate using **platform-independent and language-neutral** Web protocols.
3. A Web Service shares **schemas and contracts/interface** that can be called from another program.
4. A Web Service is **registered and can be located** through a Web Service Registry.
5. Web Services support **loosely coupled** connections between systems.

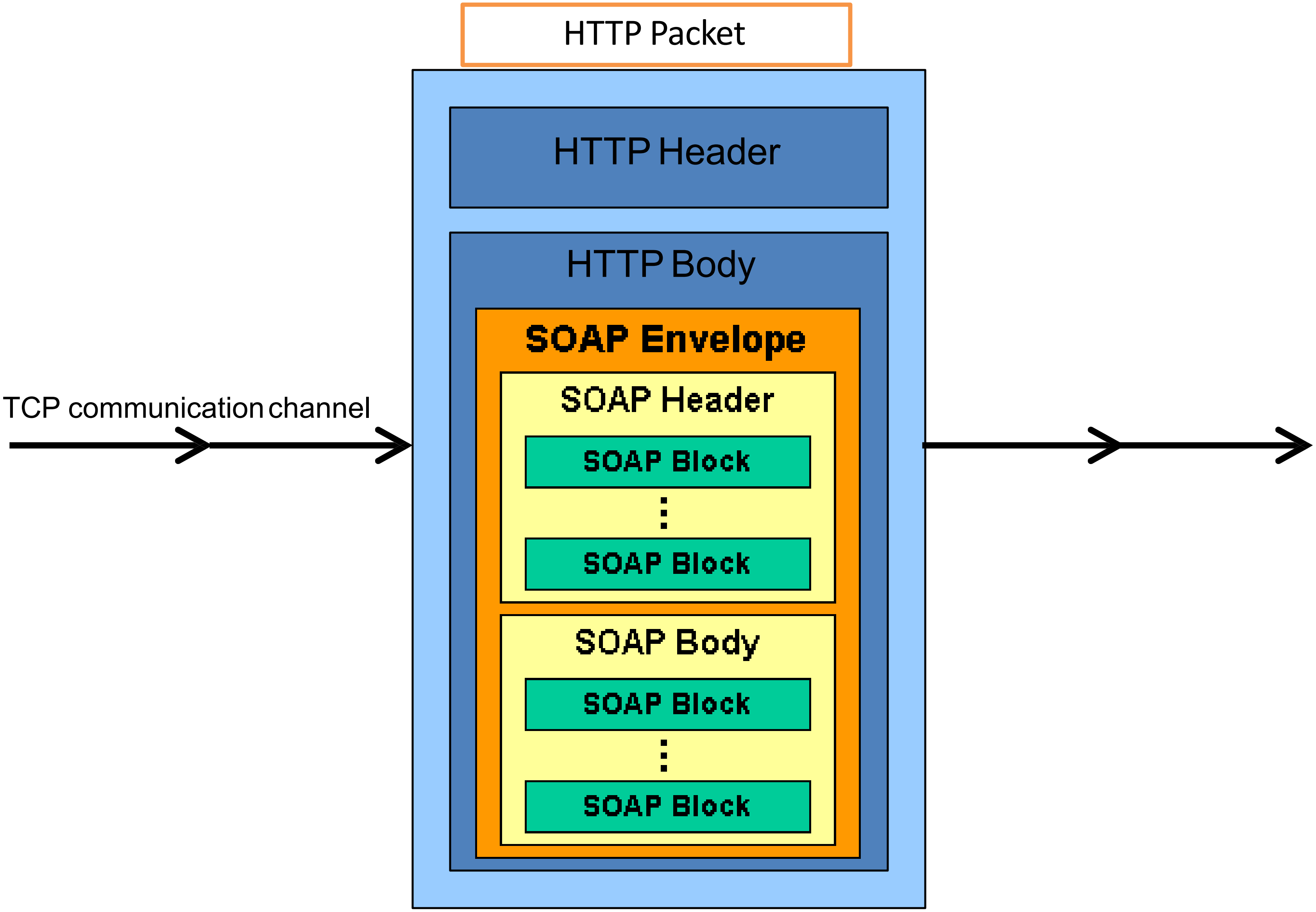
SOAP

- Simple Object Access Protocol
- SOAP is an open protocol specification defining a uniform way of performing RPCs using HTTP as the underlying communications protocol with XML for the data serialization.



PORT 80 or HTP

SOAP-Packet



HTTP Packet

TCP communication channel

HTTP Header

HTTP Body

SOAP Envelope

SOAP Header

SOAP Block

SOAP Block

SOAP Body

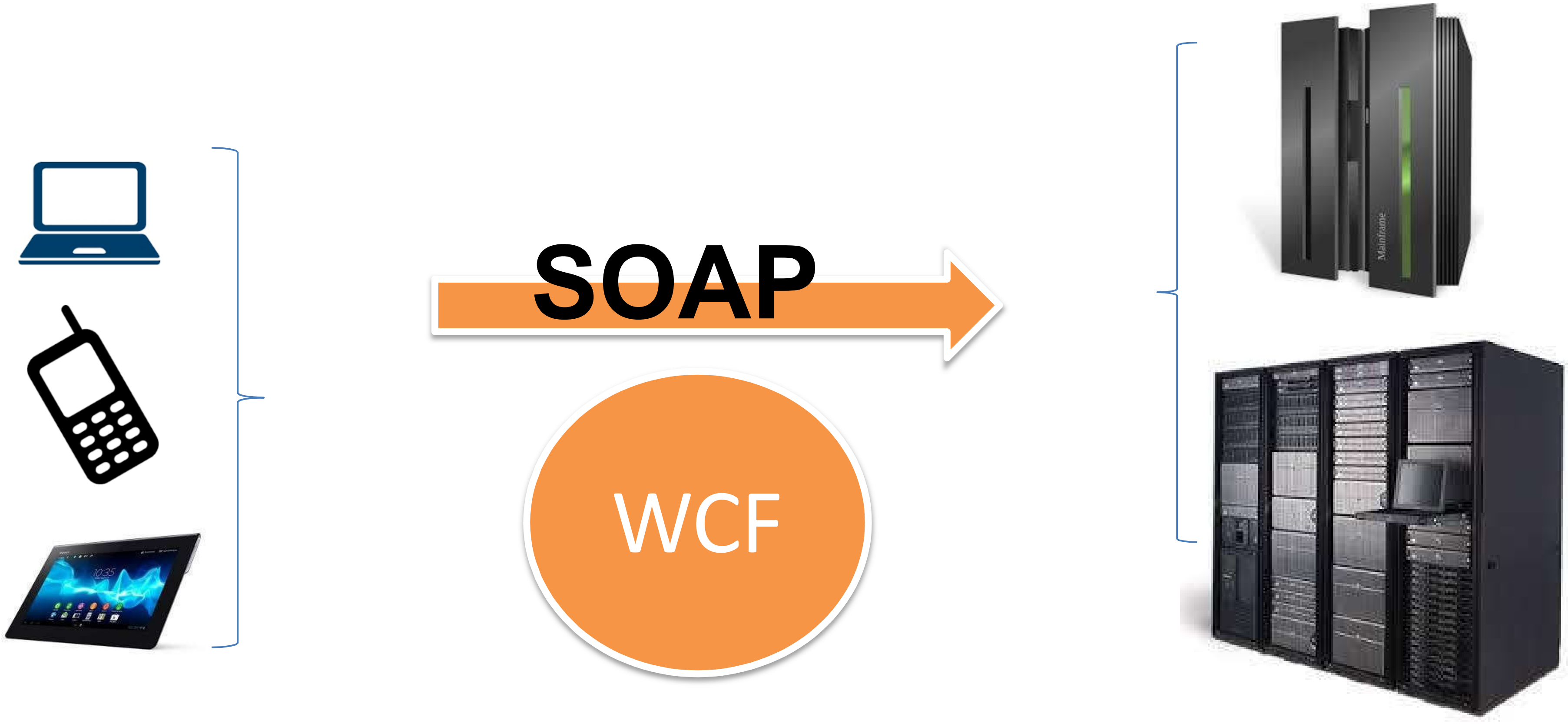
SOAP Block

SOAP Block

SOAP Messages

- SOAP provides a standard ‘envelope’ within which a message can be delivered.
- SOAP is mechanism (protocol) for transferring information (messages) between applications which may be widely distributed.
- SOAP says nothing about the content of the message – the sender and the receiver must understand the message for themselves.
- SOAP is part of a communication stack.

Web Services

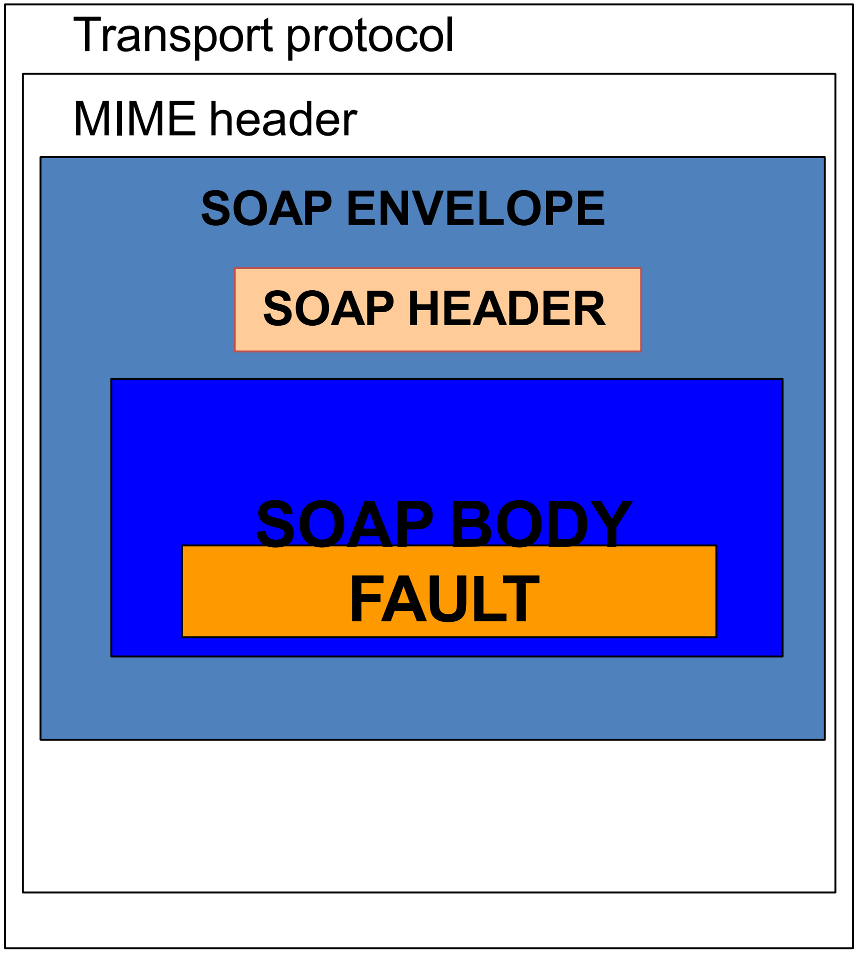


SOAP

WCF

SOAP STRUCTURE

- Each SOAP message will have:
  - An `Envelope`
  - A `Header` (optional)
  - A `Body`
  - The `Body` may contain a `Fault` element



SOAP Structure(2)

- The envelope wraps the entire soap document
- The header allows additional information to be passed as well as the body of the document – e.g. Authentication
- The body element contains the core of the SOAP document – this will contain either the RPC call or the XML message itself
- The fault information will contain any exception information

Anatomy of a SOAP message

```
<?xml version=‘1.0’
   encoding=‘UTF-8’?>
<SOAP-ENV:Envelope xmlns:SOAP_ENV=“http://schemas.xmlsoap.org/soap/envelope/”
        xmlns:xsi=“http://www.w3c.org/1999/XMLSchema-instance”
        xmlns:xsd=“http://www.w3c.org/1999/XMLSchema”>
        <SOAP-ENV:Header>

        </SOAP-ENV:Header


        <SOAP_ENV:Body>


        </SOAP-ENV:Body>


</SOAP-ENV:Envelope>
```

C#

SOAP protocol binding

```
SOAPAction = “urn:soaphttpclient-action-uri”
Host = localhost
Content-Type = text/xml; charset=utf-8
Content-Length = 701
```

```
<SOAP-ENV:Envelope xmlns:SOAP_ENV=“http://schemas.xmlsoap.org/soap/envelope/”
        xmlns:xsi=“http://www.w3c.org/1999/XMLSchema-instance”
        xmlns:xsd=“http://www.w3c.org/1999/XMLSchema”>




</SOAP-ENV:Envelope>
```

SOAP RPC

- SOAP RPC messages contain XML that represents a method call or method response

- The SOAP XML will be converted into a method call on the server and the response will be encoded into SOAP XML to be returned to the client

SOAP Faults

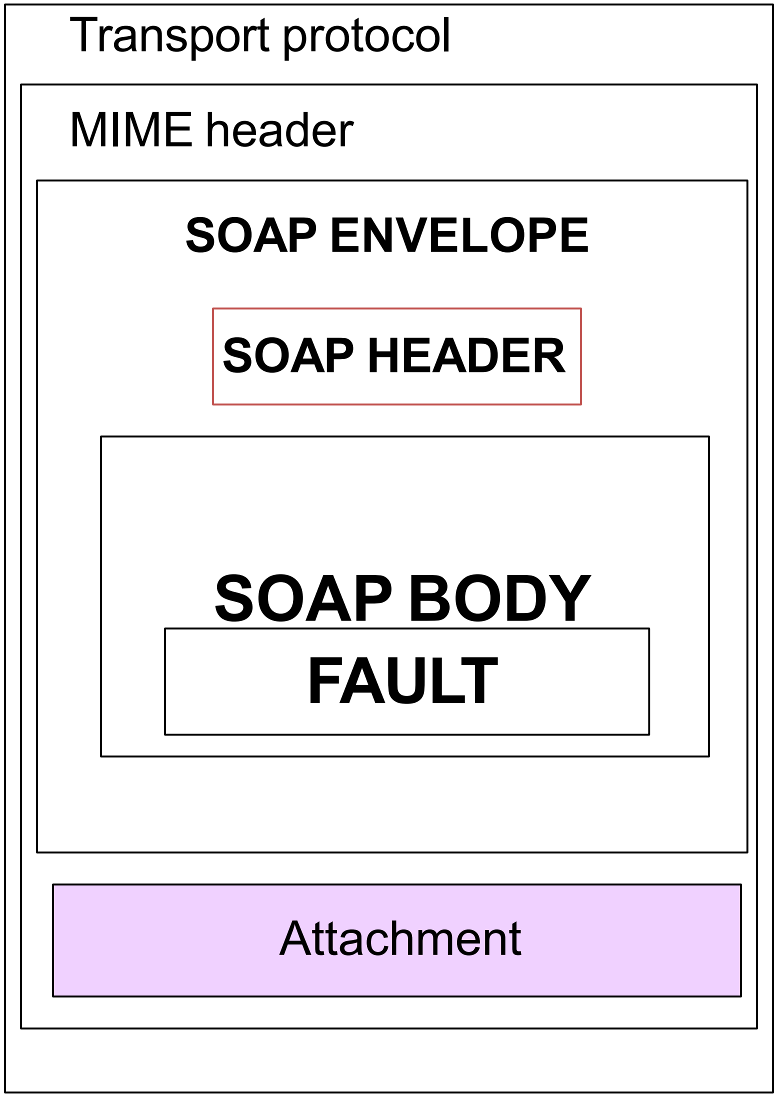
- SOAP errors are handled using a specialised envelope known as a Fault Envelope
- A SOAP Fault is a special element which must appear as an immediate child of the body element
- <faultcode> and <faultstring> are required.

A SOAP fault

```
<?xml version=‘1.0’ encoding=‘UTF-8’?>
<SOAP-ENV:Envelope
   xmlns:SOAP_ENV=“http://schemas.xmlsoap.org/soap/envelope/”
   xmlns:xsi=“http://www.w3c.org/1999/XMLSchema-instance”
   xmlns:xsd=“http://www.w3c.org/1999/XMLSchema”>
   <SOAP_ENV:Body>
            <SOAP-ENV:Fault>
                    <faultcode>SOAP-ENV:Server</faultcode>
                    <faultstring>Test fault</faultstring>
                    <faultactor>/soap/servlet/rpcrouter</faultactor>
                    <detail>
                            ..
                    </detail>
            </SOAP-ENV:Fault>



   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Attachment

- Large quantities or binary data may not fit well into a XML SOAP message.
- In which case it can be sent ‘out of band’ by attaching it to a SOAP message
- *Analogy : email attachments.*



Attaching a file to a SOAP message

- To add a file to a SOAP message a tag is added within the body of the message.

```
<?xml version=‘1.0’ encoding=‘UTF-8’?>
<SOAP-ENV:Envelope
xmlns:SOAP_ENV=“http://schemas.xmlsoap.org/soap/envelope/”
          xmlns:xsi=“http://www.w3c.org/1999/XMLSchema-instance”
          xmlns:xsd=“http://www.w3c.org/1999/XMLSchema”>
          <SOAP_ENV:Body>


                    <attachment href=“{URL}”/>


          </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP

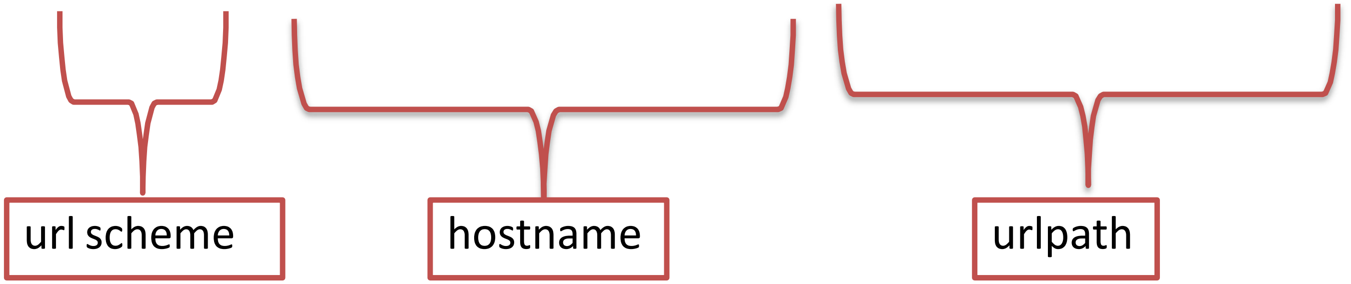
- SOAP is an XML based encoding of messages that are typically sent over HTTP, but could be sent over SMTP or even FTP
- SOAP sites on top of HTTP
- Generally require a toolkit and more processing power.

HTTP

- HTTP can serve any content over HTTP such as HTML, images, sound, video etc.
- HTTP is overTCP/IP
- HTTP based APIs refer to APIs that are exposed as one or more HTTP URIs and typical responses are in XML / JSON. Response schemas are custom per object

URL

- Uniform resource locator
  - http://www.google.com/
  - https://www.google.com/username/logo.jpg



url scheme

hostname

urlpath

http://www.askapache.com/online-tools/http-headers-tool/

Content Types

- Content type that a server specifies relies on the **Multi-purpose Internet Mail Extensions (MIME)**

Type/SubType	Description
Application/atom+xml	Atom feed
Application/json	JSON data
Image/gif	GIF image
Image/png	PNG image
Video/mp4	Mp4 video
Text/xml	Xml
Text/html	Html
Text/plain	Just text

HTTP

- Ubiquitous (common)
- Interoperable
- Scalable
- Flexible
- Mature
- Simple

Many Faces of HTTP Frameworks in .NEt

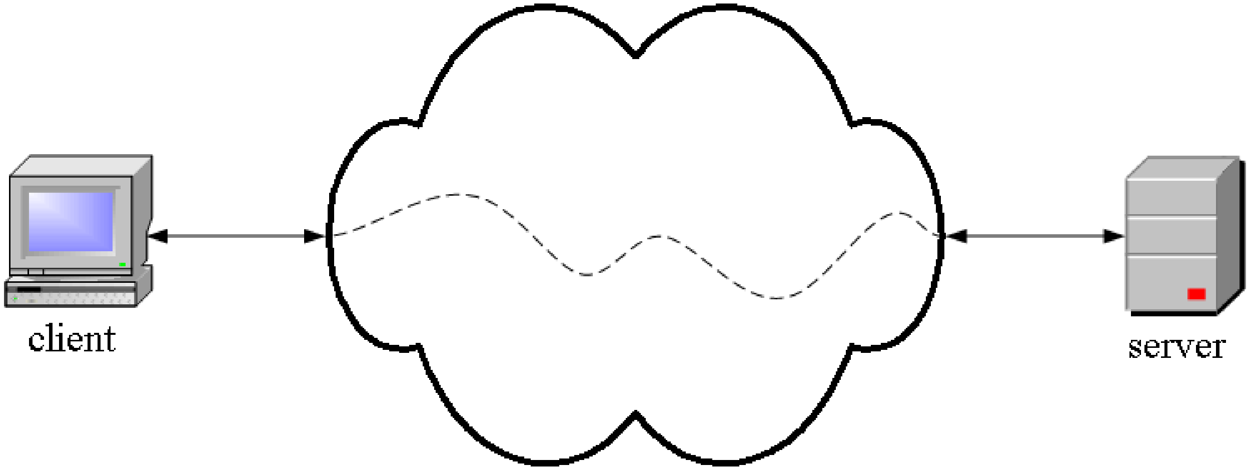
- WCF Web HTTP
- WCF Data Services
- ASP.NET MVC
- ASP.NET WEB API

HTTP

- **Stateless** – ***Each transaction between the client and server is independent and no state is set based on a previous transaction or condition.***
- Uses **requests** from the client to the server and **responses** from the server to the client for sending and receiving data.

HTTP is designed as a stateless protocol meaning each request response transaction is independent

HTTP is an application layer protocol



- The Web client and the Web server are application programs
- Application layer programs do useful work like retrieving Web pages, sending and receiving email or transferring files
- Lower layers take care of the communication details
- The client and server send messages and data without knowing anything about the communication network

HTTP

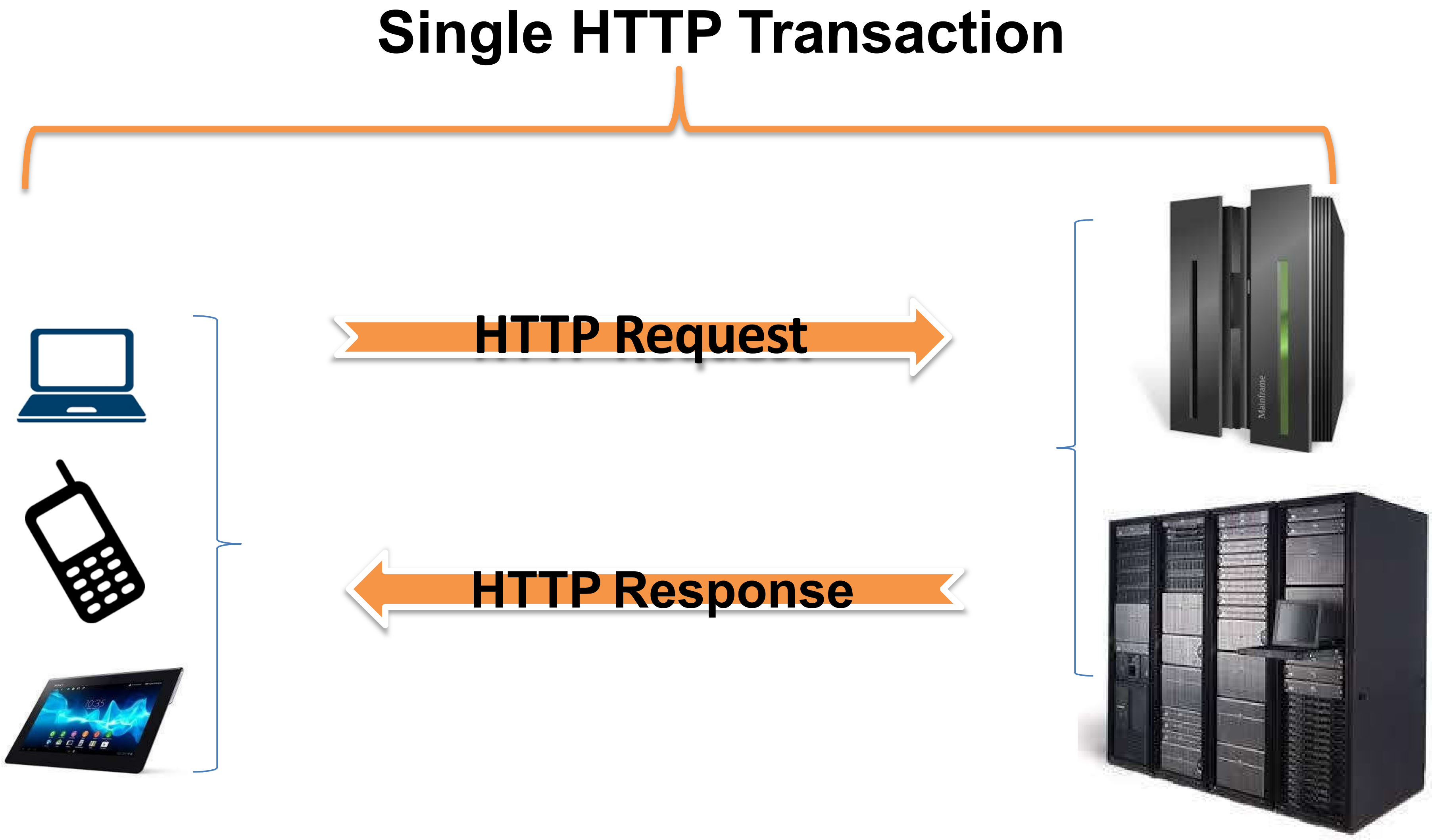
HTTP: hypertext transfer protocol

- The rules governing the conversation between a Web client and a Web server
- Request-response protocol
- It is a stateless (does not maintain a state of a session) and asynchronous(an html document is loaded asynchronous by the browser as soon as parts of it are available)

Layer	Function
Application	Do useful work like Web browsing, email, and file transfer
Lower layers	Handle communication between the client and server

A network protocol is the set of rules governing a conversation between a client and a server

HTTP Message Types

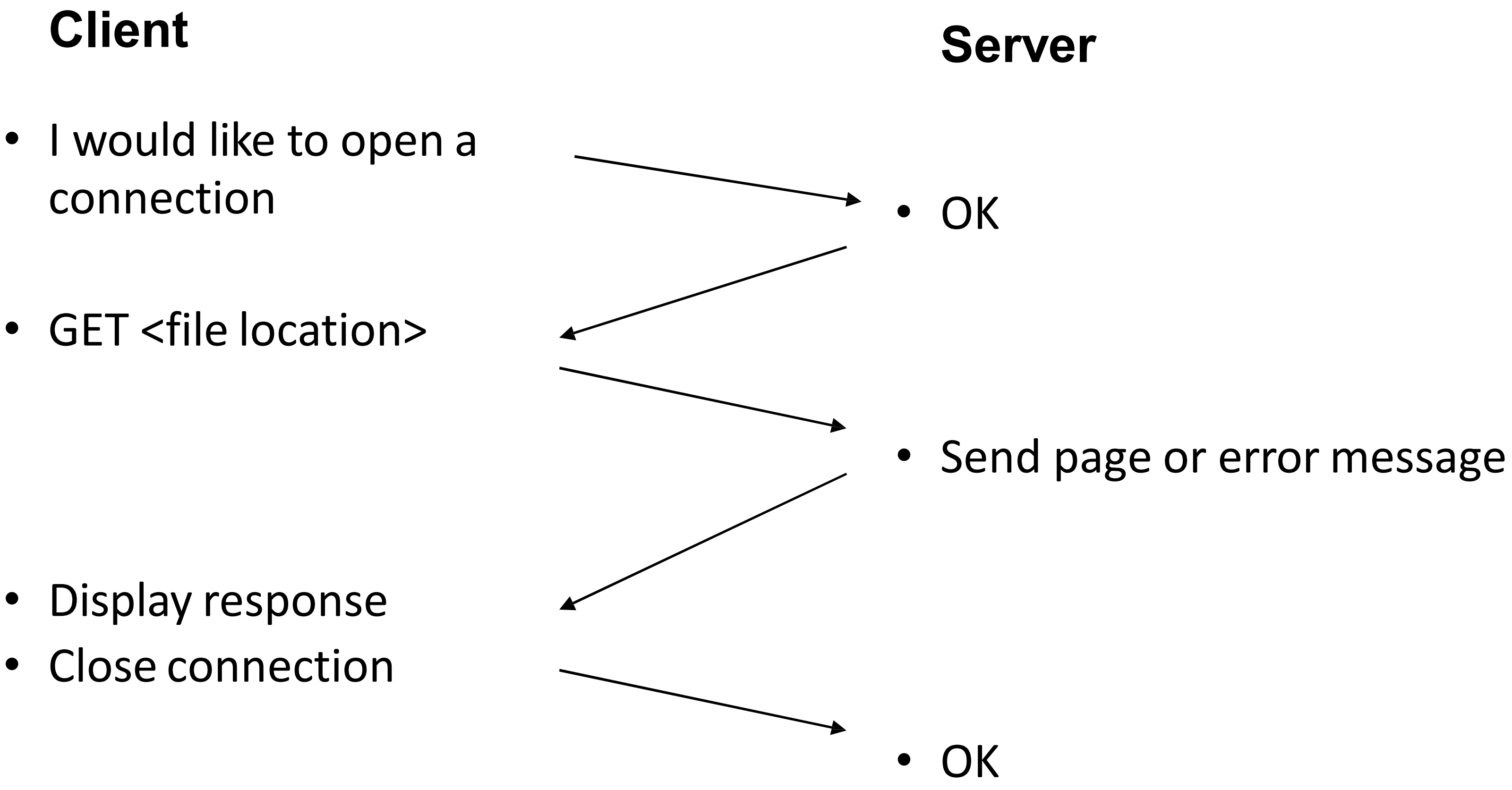


Single HTTP Transaction

HTTP Request

HTTP Response

An HTTP conversation



Client

Server

- I would like to open a connection

- OK

- GET <file location>

- Send page or error message

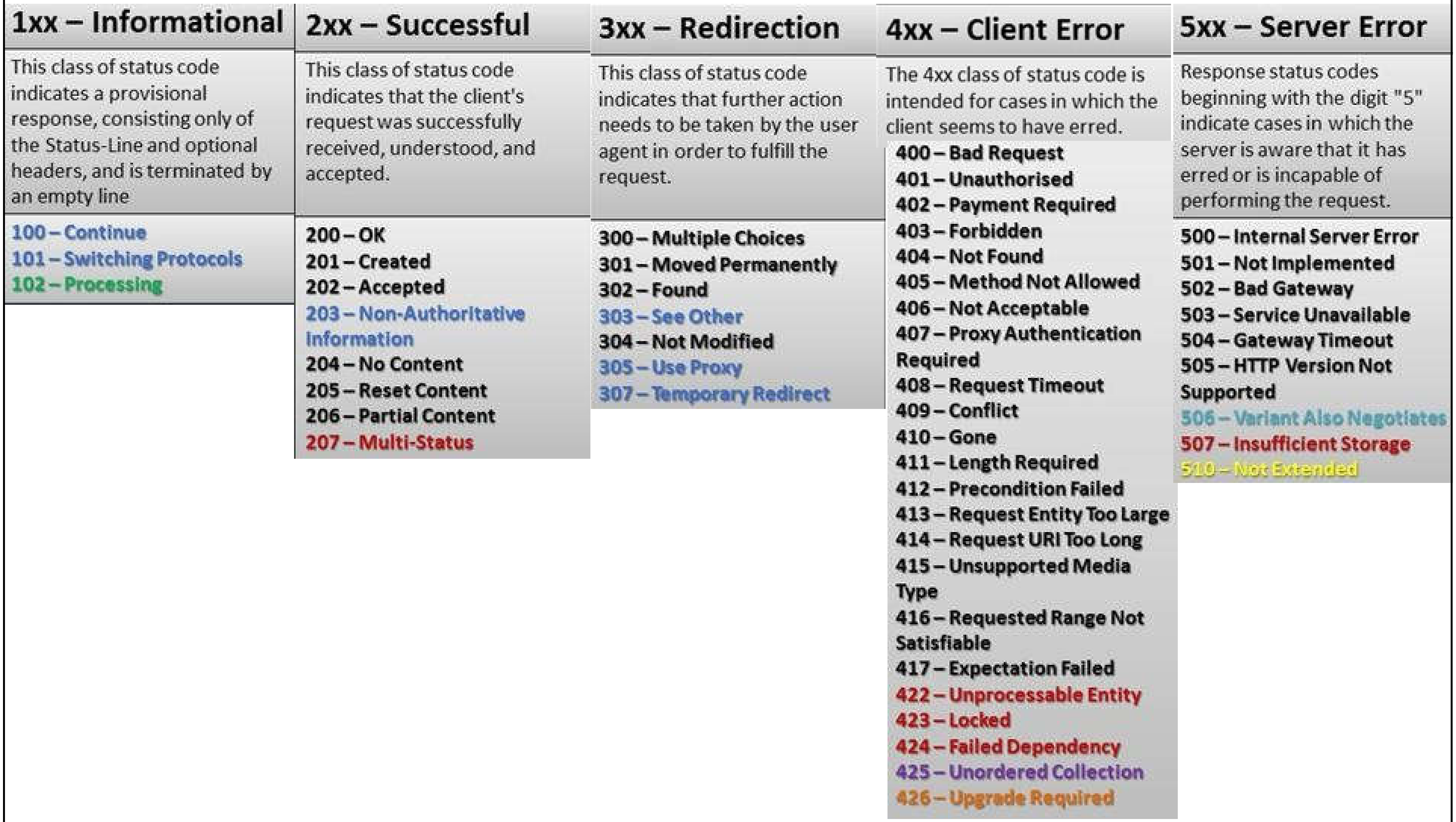
- Display response
- Close connection

- OK

HTTP is the set of rules governing the format and content of the conversation between a Web client and server

C#

HTTP Status Codes



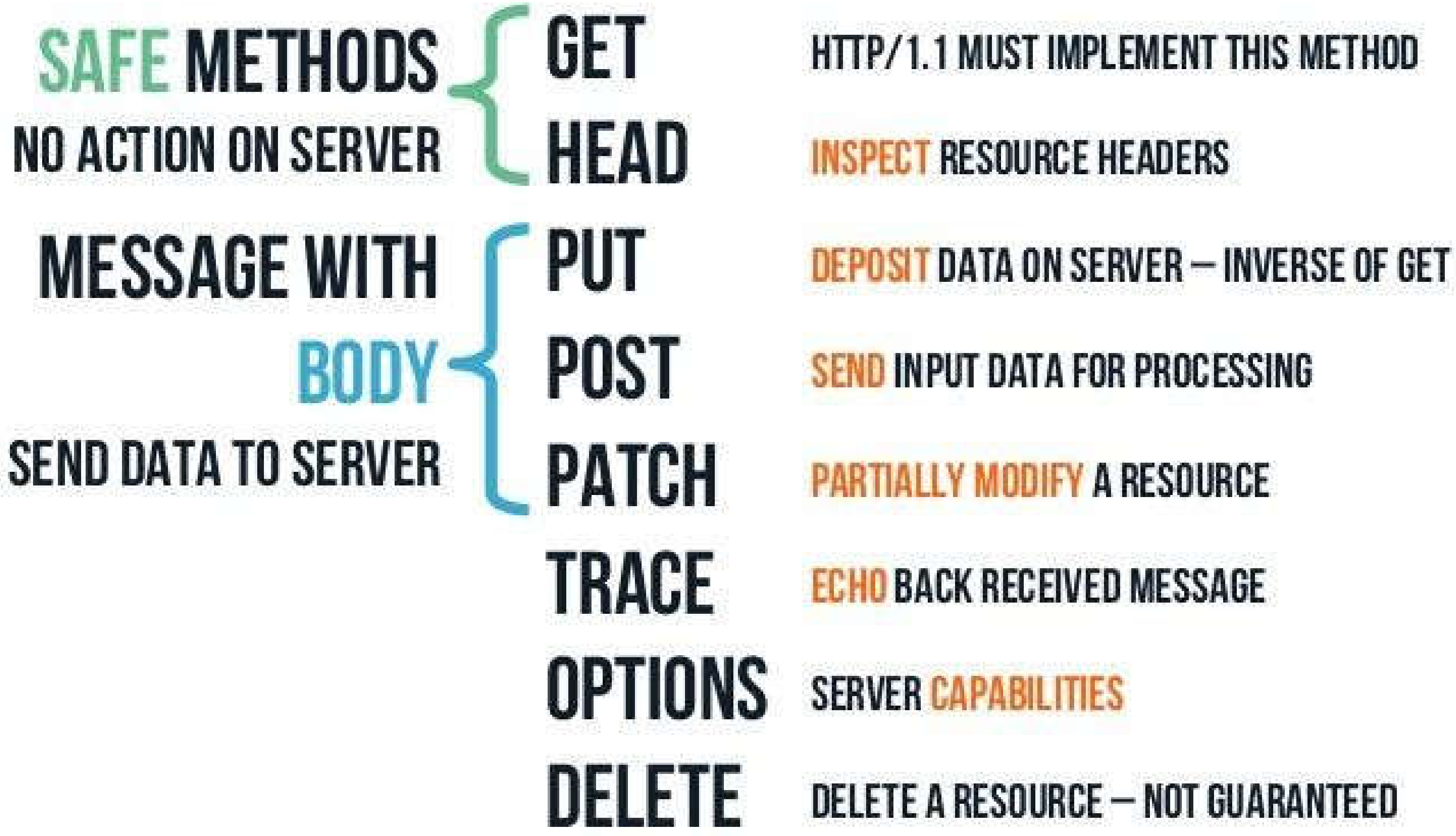
1xx – Informational	2xx – Successful	3xx – Redirection	4xx – Client Error	5xx – Server Error
This class of status code indicates a provisional response, consisting only of the Status-Line and optional headers, and is terminated by an empty line	This class of status code indicates that the client's request was successfully received, understood, and accepted.	This class of status code indicates that further action needs to be taken by the user agent in order to fulfill the request.	The 4xx class of status code is intended for cases in which the client seems to have erred.	Response status codes beginning with the digit "5" indicate cases in which the server is aware that it has erred or is incapable of performing the request.
100 – Continue 101 – Switching Protocols 102 – Processing	200 – OK 201 – Created 202 – Accepted 203 – Non-Authoritative Information 204 – No Content 205 – Reset Content 206 – Partial Content 207 – Multi-Status	300 – Multiple Choices 301 – Moved Permanently 302 – Found 303 – See Other 304 – Not Modified 305 – Use Proxy 307 – Temporary Redirect	400 – Bad Request 401 – Unauthorised 402 – Payment Required 403 – Forbidden 404 – Not Found 405 – Method Not Allowed 406 – Not Acceptable 407 – Proxy Authentication Required 408 – Request Timeout 409 – Conflict 410 – Gone 411 – Length Required 412 – Precondition Failed 413 – Request Entity Too Large 414 – Request URI Too Long 415 – Unsupported Media Type 416 – Requested Range Not Satisfiable 417 – Expectation Failed 422 – Unprocessable Entity 423 – Locked 424 – Failed Dependency 425 – Unordered Collection 426 – Upgrade Required	500 – Internal Server Error 501 – Not Implemented 502 – Bad Gateway 503 – Service Unavailable 504 – Gateway Timeout 505 – HTTP Version Not Supported 506 – Variant Also Negotiates 507 – Insufficient Storage 510 – Not Extended

HTTP Request

Request-Method is:

- **GET** – request whatever information is identified by the Request-URL
- **POST** – request that server accepts the entity enclosed in the request
- **OPTIONS** - request for information about communication options
- **PUT** – request that the enclosed entity be stored under the Request-URL
- **DELETE** – request that the server delete the resource identified by Request-URL
- **TRACE** – invoke a remote, application-layer loopback of the request message
- **CONNECT** – used by proxies in SSL connections
- **HEAD** – identical to GET, but server must not return a message body in response

- Request-header can have the following fields (selection):
  - Accept : MIME types of resources accepted by browser
  - Accept-Charset : charset accepted by browser
  - Accept-Encoding : encoding accepted by browser
  - Accept-Language : language accepted by browser
  - Authorization : user-agent wishes to authenticate itself with a server
  - Host : the host Request-URL points to
  - Referer : the URL of document refering this URL
  - User-Agent : Firefox, Safari, IE



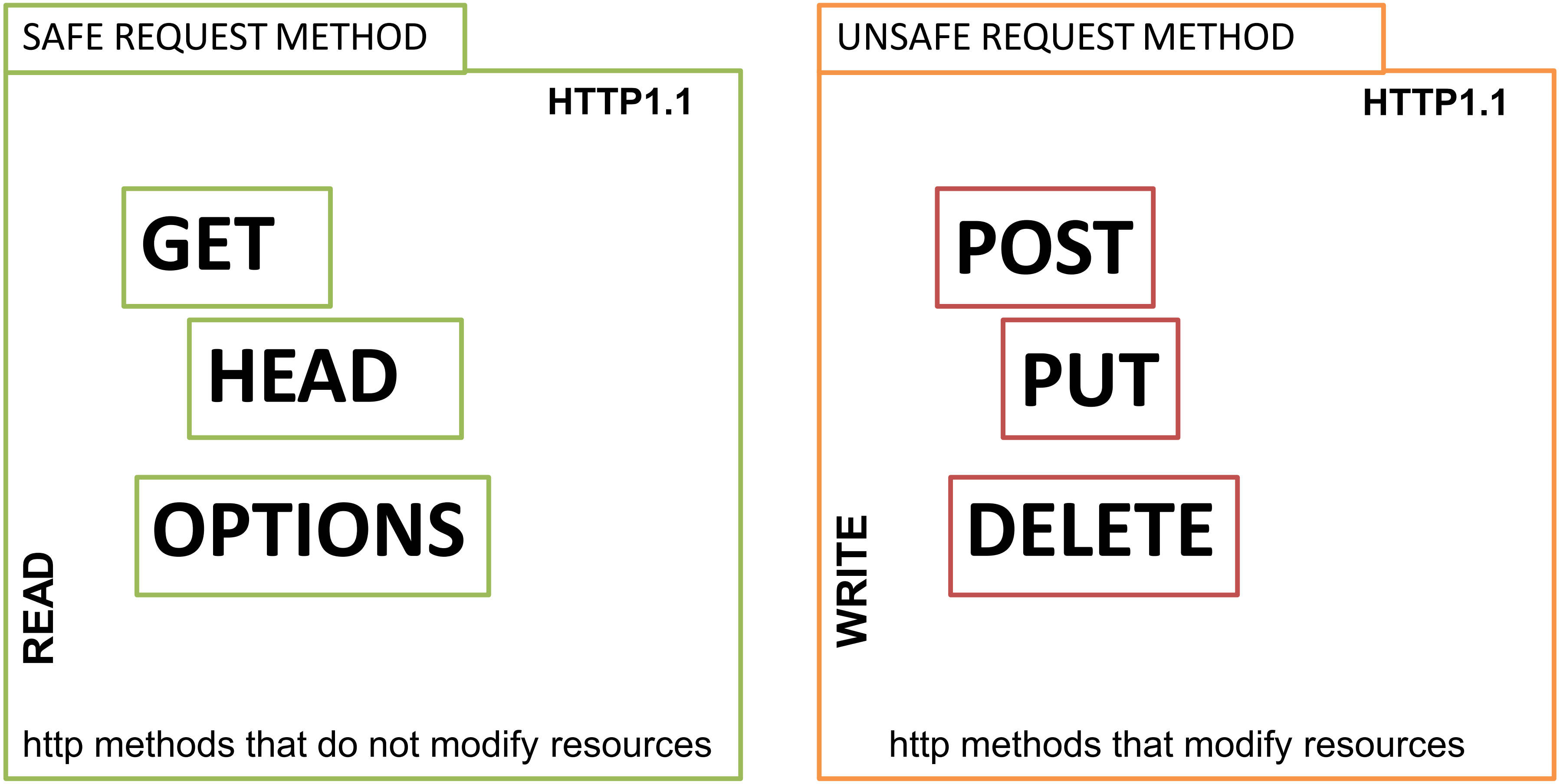
SAFE METHODS
NO ACTION ON SERVER

MESSAGE WITH BODY
SEND DATA TO SERVER

GET	HTTP/1.1 MUST IMPLEMENT THIS METHOD
HEAD	INSPECT RESOURCE HEADERS
PUT	DEPOSIT DATA ON SERVER – INVERSE OF GET
POST	SEND INPUT DATA FOR PROCESSING
PATCH	PARTIALLY MODIFY A RESOURCE
TRACE	ECHO BACK RECEIVED MESSAGE
OPTIONS	SERVER CAPABILITIES
DELETE	DELETE A RESOURCE – NOT GUARANTEED

C#

Safe vs Unsafe HTTP Request Methods



SAFE REQUEST METHOD

HTTP1.1

GET

HEAD

OPTIONS

READ

http methods that do not modify resources

UNSAFE REQUEST METHOD

HTTP1.1

POST

PUT

DELETE

WRITE

http methods that modify resources

HTTP vs HTTPS

HTTP

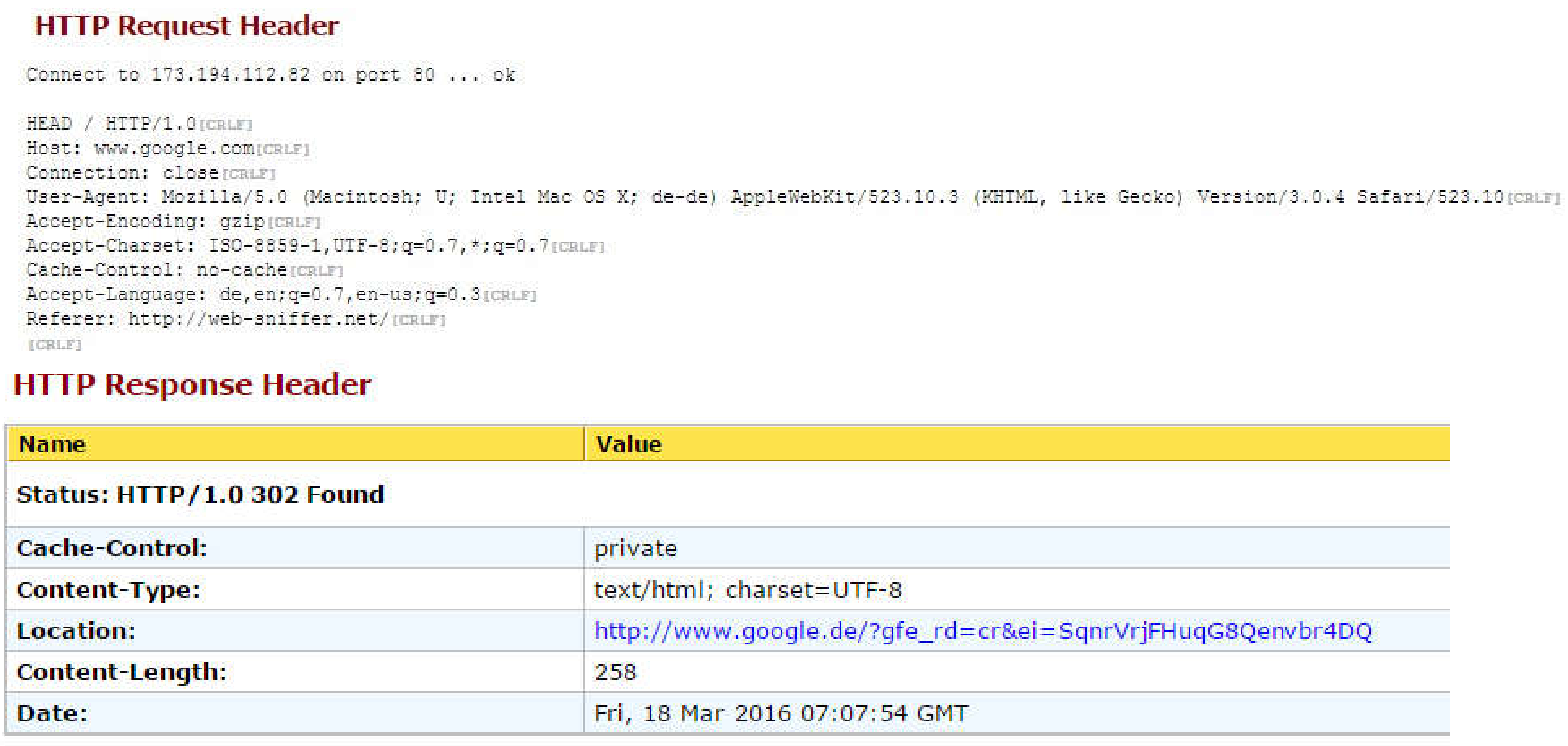
- It is hypertext transfer protocol
- It is not secure and unreliable
- HTTP urls begin with http://
- It uses port 80 by default
- It is subject to man-in-the-middle and eavesdropping attacks

HTTPS

- It is hypertext transfer protocol with secure
- It is secure and reliable
- Https urls being with https
- It uses port 443 by default
- It is designed to withstand such attacks and is considered secure against such attacks

http://www.rexswain.com/httpview.html

http://headers.cloxy.net/



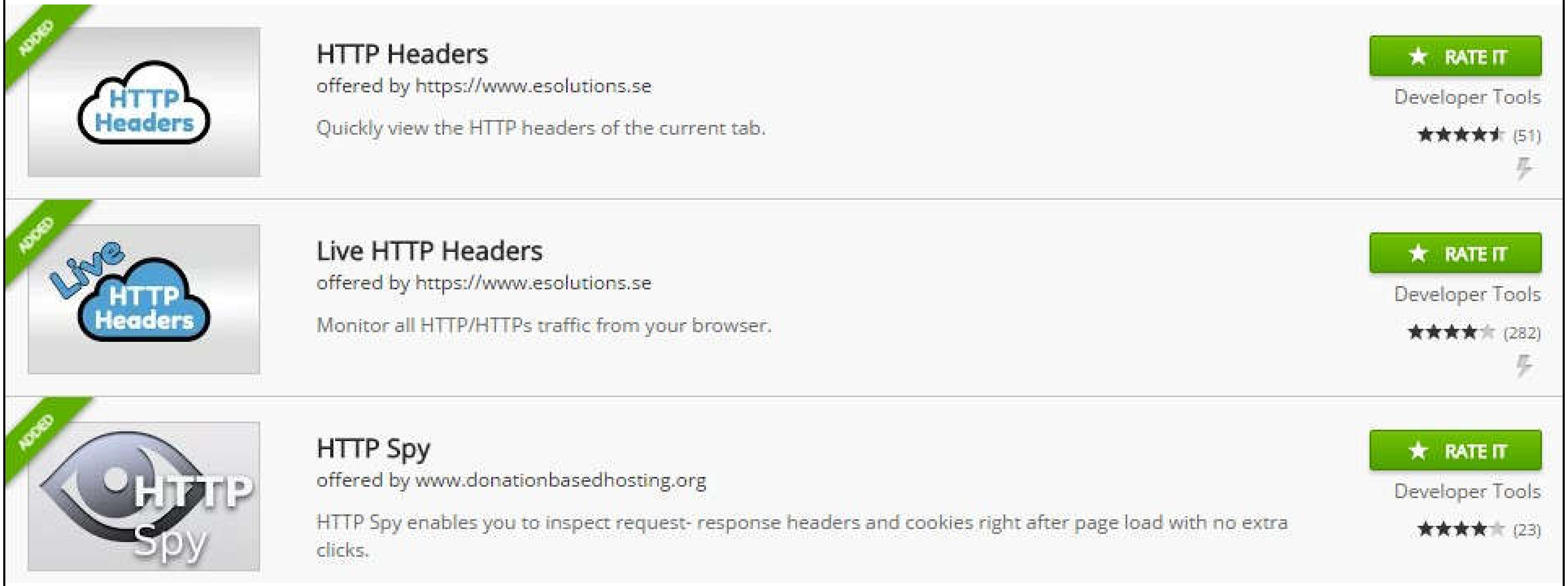
HTTP Request Header

```
Connect to 173.194.112.82 on port 80 ... ok

HEAD / HTTP/1.0[CRLF]
Host: www.google.com[CRLF]
Connection: close[CRLF]
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X; de-de) AppleWebKit/523.10.3 (KHTML, like Gecko) Version/3.0.4 Safari/523.10[CRLF]
Accept-Encoding: gzip[CRLF]
Accept-Charset: ISO-8859-1,UTF-8;q=0.7,*;q=0.7[CRLF]
Cache-Control: no-cache[CRLF]
Accept-Language: de,en;q=0.7,en-us;q=0.3[CRLF]
Referer: http://web-sniffer.net/[CRLF]
[CRLF]
```

HTTP Response Header

Name	Value
Status: HTTP/1.0 302 Found	
Cache-Control:	private
Content-Type:	text/html; charset=UTF-8
Location:	http://www.google.de/?gfe_rd=cr&ei=SqnrVrjFHuqG8Qenvbr4DQ
Content-Length:	258
Date:	Fri, 18 Mar 2016 07:07:54 GMT

Chrome add on tools for HTTP Header viewing



HTTP Headers

offered by https://www.esolutions.se

Quickly view the HTTP headers of the current tab.

Developer Tools

★★★★★ (51)

Live HTTP Headers

offered by https://www.esolutions.se

Monitor all HTTP/HTTPs traffic from your browser.

Developer Tools

★★★★☆ (282)

HTTP Spy

offered by www.donationbasedhosting.org

HTTP Spy enables you to inspect request- response headers and cookies right after page load with no extra clicks.

Developer Tools

★★★★☆ (23)

Request Messages

```
[method] [URL] [version]
[headers]
[body]
```

```
GET http://www.sycliq.com/articles/index.aspx
Host: google.com
Accept-Language : en-EN
Date: FRI, 19 Jan, 2016 10:10:26 GMT
```

Common Request Headers

Header	Description
Referer	The URL of the referring page
User-Agent	Information about the browser
Accept	Preferred media types
Accept-Language	Preferred Language
Cookie	Cookie information
If-Modified-Since	Date of last retrieval
Date	Creation timestamp for the message

Full Request

```
Get/HTTP/1.1
Host: sycliq.com
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows 10; WOW64) Chrome/16.0.912.75
Accept: text/html, application/xhtml+xml, application/xml;q=0.9,*/*; q=0.8
Referer:  http://www.google.com/url?&q=iot
Accept-Encoding: gzip, defalte, sdch
Accept-Language: en-Us, en; q=0.8
Accept-Charset: ISO-8859-1, utf-8;q=0.7,*q=0.3
```

http://www.askapache.com/online-tools/http-headers-tool/

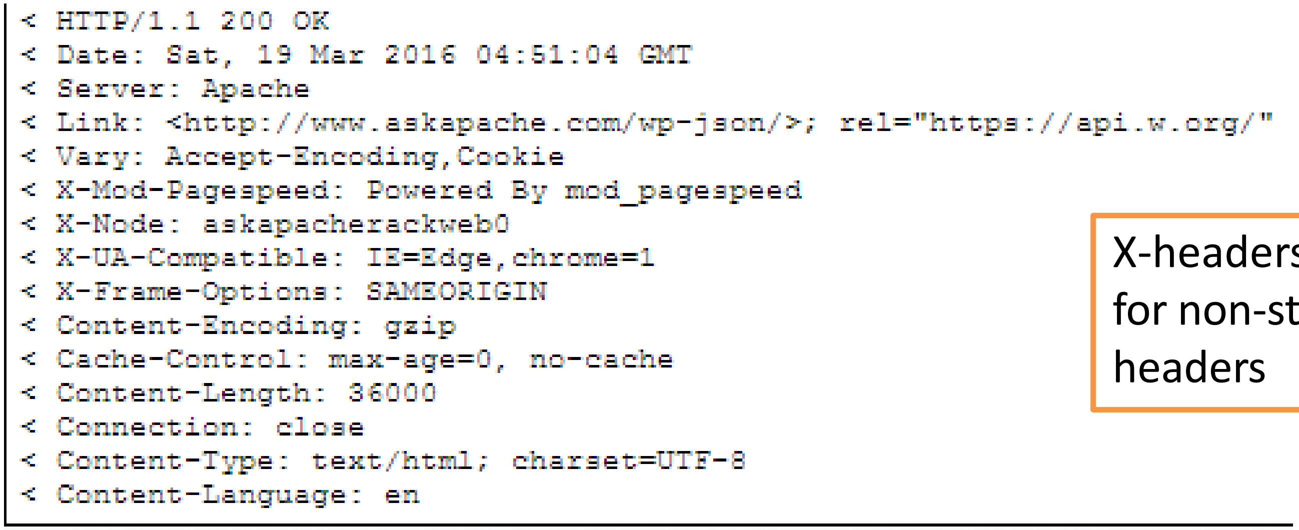
HTTP RESPONSE

https://curlbuilder.com/

[version] [status][reason]

[headers]

[body]



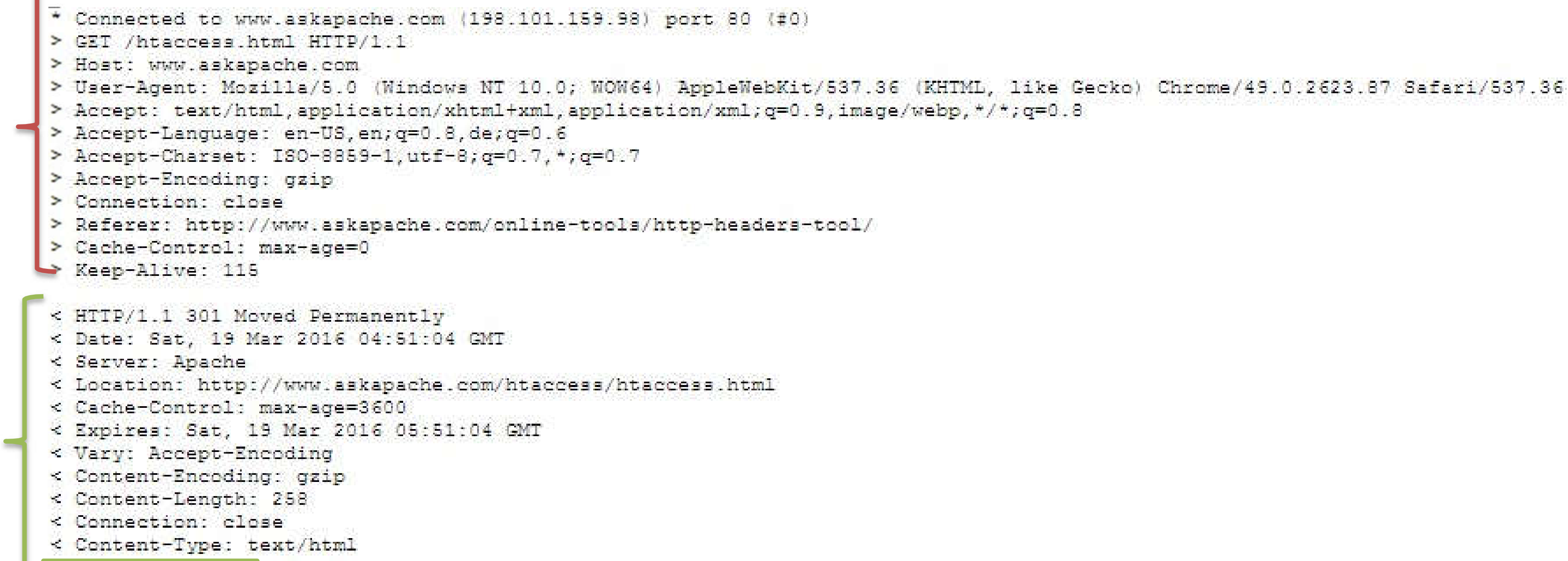
```
< HTTP/1.1 200 OK
< Date: Sat, 19 Mar 2016 04:51:04 GMT
< Server: Apache
< Link: <http://www.askapache.com/wp-json/>; rel="https://api.w.org/"
< Vary: Accept-Encoding,Cookie
< X-Mod-Pagespeed: Powered By mod_pagespeed
< X-Node: askapacherackweb0
< X-UA-Compatible: IE=Edge,chrome=1
< X-Frame-Options: SAMEORIGIN
< Content-Encoding: gzip
< Cache-Control: max-age=0, no-cache
< Content-Length: 36000
< Connection: close
< Content-Type: text/html; charset=UTF-8
< Content-Language: en
```

X-headers are reserved for non-standard headers

C#

HTTP Header

Request



```
* Connected to www.askapache.com (198.101.159.98) port 80 (#0)
> GET /htaccess.html HTTP/1.1
> Host: www.askapache.com
> User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.87 Safari/537.36
> Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
> Accept-Language: en-US,en;q=0.8,de;q=0.6
> Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
> Accept-Encoding: gzip
> Connection: close
> Referer: http://www.askapache.com/online-tools/http-headers-tool/
> Cache-Control: max-age=0
> Keep-Alive: 115

< HTTP/1.1 301 Moved Permanently
< Date: Sat, 19 Mar 2016 04:51:04 GMT
< Server: Apache
< Location: http://www.askapache.com/htaccess/htaccess.html
< Cache-Control: max-age=3600
< Expires: Sat, 19 Mar 2016 05:51:04 GMT
< Vary: Accept-Encoding
< Content-Encoding: gzip
< Content-Length: 258
< Connection: close
< Content-Type: text/html
```

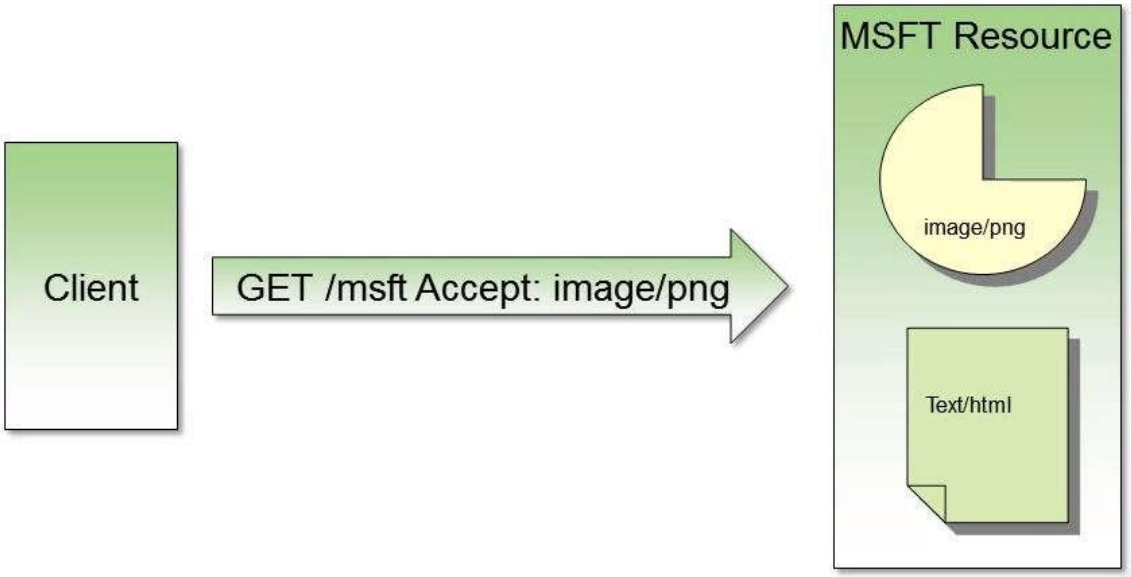
Response

Content Negotiation

- It is the process of selecting the best representation for a given response when there are multiple representations available.
- Content negotiation (conneg) in the ASP.NET Web API is an intrinsic server-driven mechanism used to determine, based on the client's request, which media type formatter (out of the box there are 4 media type formatters) to be used to return an API response.
- In general a client sends the Accept parameter in the Request Header to determine the response.
- In .NET, it really comes down to deciding how to send down your CLR object to the client, over HTTP or from the ASP.NET Web API perspective, serialization is the process of translating a .NET Common Language Runtime (CLR) type into a format that can be transmitted over HTTP. The default formats are either JSON or XML.

Content Negotiation

- Core mechanism of HTTP
- Client specific desired formats using AcceptHeader

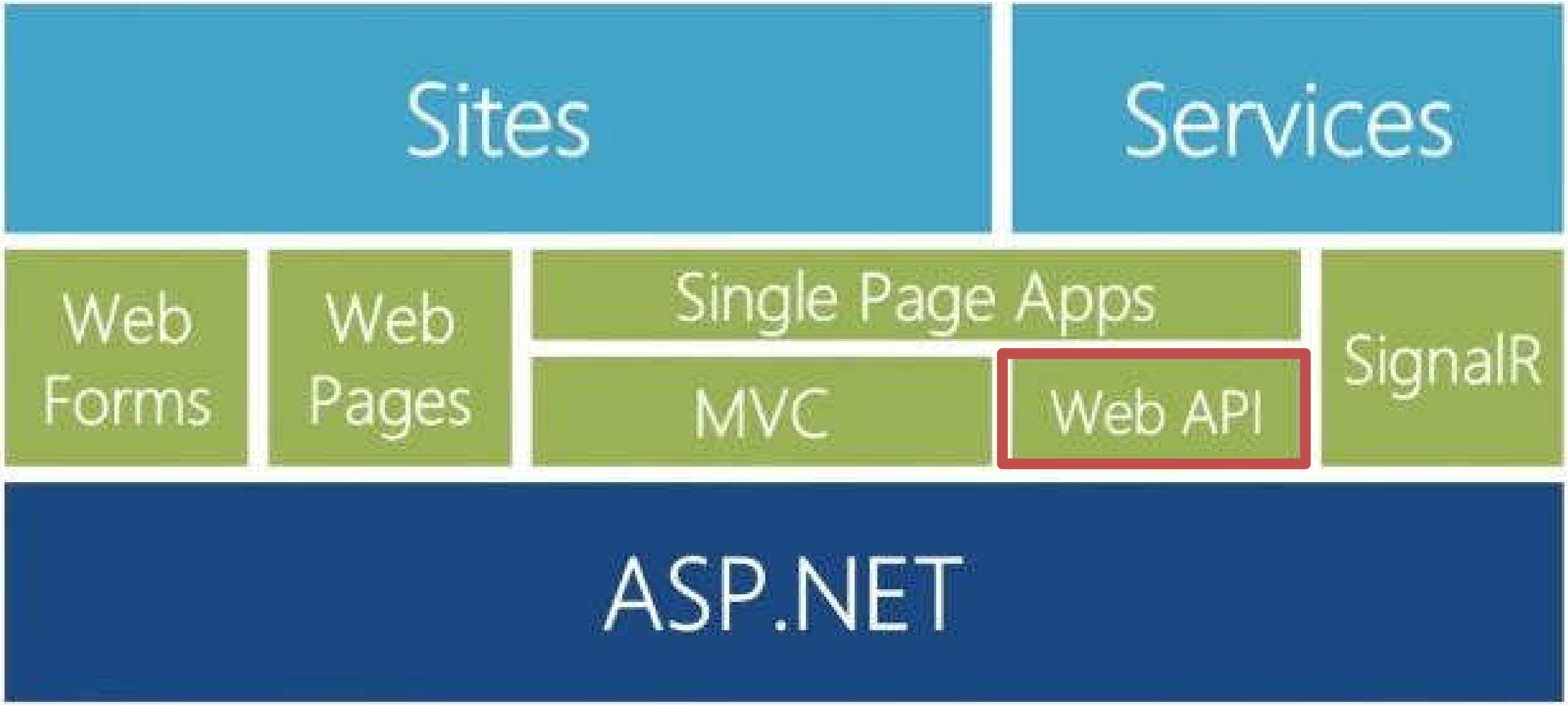


Summary

- We understood the evolution of web services to support cross platform applications using light weight http application protocol.

ASP.NET WEB API

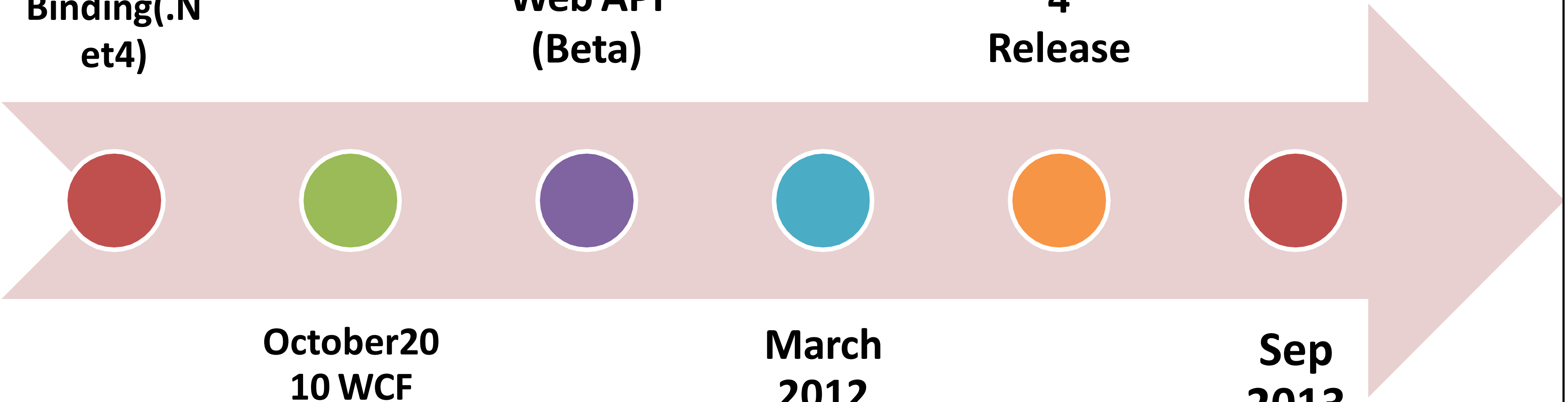
ASP.NET


Sites
Services
Web Forms
Web Pages
Single Page Apps
MVC
Web API
SignalR
ASP.NET

ASP.NET WEB API

A Framework for creating HTTP Services that can reach a broad range of clients including browsers and mobile devices

ASP.NET WEB API TIMELINE



April 2010 WCF WebHttp Binding(.Net4)

October2010 WCF Web API on CodePlex

Feb 2012 ASP.Net Web API (Beta)

March 2012 ASP.Net OpenSource

Aug 2012 ASP.Net Web API 4 Release

Sep 2013 ASP.Net Web API 2 RC

HTTP & ASP.NET WEB API

- HTTP Fundamentals via Web API
  - HTTP Messages
  - URIs
  - Verbs
  - Controllers and Actions
  - Status Code
  - HttpRequestMessage
  - HttpResponseMessage
  - Error Handling
  - Content Negotiation
  - MediaType Formatters
  - OData
  - Validations
  - Dependency Resolver
- Hosting
  - HTTP.SYS
  - IIS 101
  - HTTP Compression
  - Persisted Connections
  - Web API Self Hosting
- More HTTP and Web API
  - Caching
  - Concurrency
  - Security
  - Streaming
  - WebSockets and SignalR

What?

- A fully supported and extensible framework for building HTTP based endpoints
- Built on the top of ASP.NET
  - Mostly ASP.NET Routing
- Released with ASP.NET MVC4
  - Not linked to MVC – you can use alone, with MVC4 or you can use with ASP.NET web forms
  - Available via NuGET
- Also includes a new HTTP Client

Why?

- First-class modern HTTP programming model
- Easily map resources to URIs and implement the uniform interface of HTTP
- Rich support for formats and HTTP content negotiation
- Request validation
- Enable hypermedia with link generations
- Separate out cross cutting concerns (like authorization, caching)
- Help Page generation
- Flexible hosting
- Light-weight testable, scales

Why?

- You are building
  - An HTML5 application that needs a services layer
  - A mobile application that needs a services layer
  - A client-server desktop application that needs a services layer
  - Reach more clients (Native Mobile Applications, Cross platform applications)
  - Scale with the cloud
  - Embrace HTTP as an Application protocol

Why to choose Web API?

- If we need a Web Service and don’t need SOAP, then ASP.Net Web API is best choice.
- It is used to build simple, non-SOAP-based HTTP Services on top of existing WCF message pipeline.
- It doesn't have tedious and extensive configuration like WCF REST service.
- Simple service creation with Web API. With WCF REST Services, service creation is difficult.
- It is only based on HTTP and easy to define, expose and consume in a REST-ful way.
- It is light weight architecture and good for devices which have limited bandwidth like smart phones.
- It is open source.

HTTP & ASP.NET WEB API

- HTTP Fundamentals via Web API
  - HTTP Messages
  - URIs
  - Verbs
  - Controllers and Actions
  - Status Code
  - HttpRequestMessage
  - HttpResponseMessage
  - Error Handling
  - Content Negotiation
  - MediaType Formatters
  - OData
  - Validations
  - Dependency Resolver
- Hosting
  - HTTP.SYS
  - IIS 101
  - HTTP Compression
  - Persisted Connections
  - Web API Self Hosting
- More HTTP and Web API
  - Caching
  - Concurrency
  - Security
  - Streaming
  - WebSockets and SignalR

What’s new?

ASP.NET Web API 2

- Attribute Routing
- Improved testability (IHTTPActionResult, HttpRequestContext)
- Odata: $select, $expand, $value,$batch
- Request batching
- OWIN (Open Web Interface for .NET) Integration
- Portable Web API Clients
- Web API Security (CORS,OAuth2.0, Authentication filters, filter overrides)

ASP.NET Web 2.1

- Attribute Routing improvements
- Global error handling
- Help page improvements
- IgnoreRoute support
- BSON formatter
- Better async filter
- Portable query building and parsing

WEB API Features

- It supports convention-based CRUD Actions since it works with HTTP verbs GET,POST,PUT and DELETE.
- Responses have an Accept header and HTTP status code.
- Responses are formatted by Web API's MediaTypeFormatter into JSON, XML or whatever format you want to add as a MediaTypeFormatter.
- It may accepts and generates the content which may not be object oriented like images, PDF files etc.
- It has automatic support for OData. Hence by placing the new [Queryable] attribute on a controller method that returns IQueryable, clients can use the method for OData query composition.
- It can be hosted with in the application or on IIS.
- It also supports the MVC features such as routing, controllers, action results, filter, model binders, IOC container or dependency injection that makes it more simple and robust.

ASP.NET WEB API TOOLS

- ASP.NET WEB API 2 Ships with Visual Studio 2013
- Requires .NET 4.5 or later
- http://www.asp.net/web-api

- Fiddler
- Getpostman (chrome addon)
- XHR POSTER (chrome addon)
- HTTPHeaders (chrome addon)

https://curlbuilder.com/

http://jflasher.github.io/spark-helper/

https://www.hurl.it/

Is this REST?

- The ASP.NET Web API doesn’t dictate an architectural style
- However you can build a RESTful service on top of it
  - It does not get in your way if you want to design using the REST architectural style

Versus the WCF Web Framework

- WCF also has a framework for building HTTP based services
  - Based on WCF attributed programming and configuration models
- When would you choose WCF over ASP.NET Web API
  - You are limited to .NET 3.5
  - You also are exposing a SOAP based services

WCF Challenges

- 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.
- To use WCF as WCF Rest service you have to enable webHttpBindings.
- It support HTTP GET and POST verbs by [WebGet] and [WebInvoke] attributes respectively.
- To enable other HTTP verbs you have to do some configuration in IIS to accept request of that particular verb on .svc files
- Passing data through parameters using a WebGet needs configuration. The UriTemplate must be specified
- It support XML, JSON and ATOM data format.

Choosing between WCF or WEB API

- Choose WCF when you want to create a service that should support special scenarios such as one way messaging, message queues, duplex communication etc.
- Choose WCF when you want to create a service that can use fast transport channels when available, such as TCP, Named Pipes, or maybe even UDP (in WCF 4.5), and you also want to support HTTP when all other transport channels are unavailable.
- Choose Web API when you want to create a resource-oriented services over HTTP that we can use the full features of HTTP (like URIs, request/response headers, caching, versioning, various content formats).
- Choose Web API when you want to expose your service to a broad range of clients including browsers, mobiles, iphone and tablets.

MVC

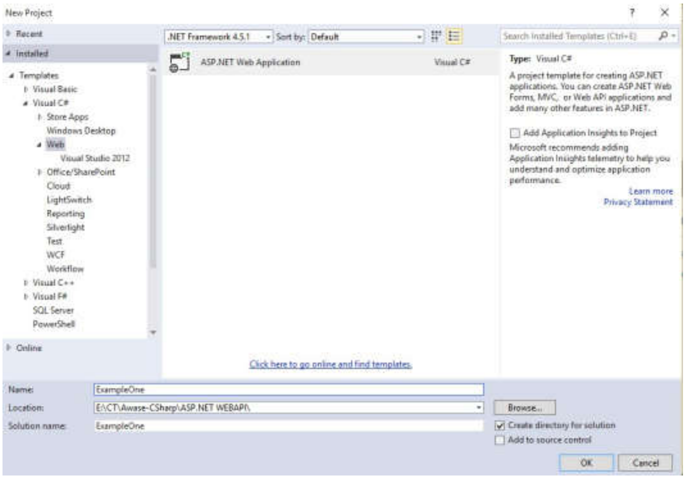
- Used to create web applications that comply with MVC design patterns generating both views and data
- MVC has provision to return data in JSON format using JsonResult
- Requests are mapped on to the action name.
- Model binding, filter, router and other MVC features are different from ASP.NET and extends from system.web.mvc

Web API

- Used to create full blown HTTP services with easy and simple way that returns only data
- Helps to build REST-ful services over the .NET framework and it also supports content-negotiation (deciding the best response format data that could be acceptable by the client ex: JSON, XML, etc.), Self hosting
- Requests are mapped to the actions based on HTTP verbs
- Web API can be used with ASP.NET and as a stand alone service layer and extends from system.web.http assembly
- A light weight architecture

INTRODUCTION, ROUTING, ATTRIBUTE ROUTING

Getting Started



New Project

- Recent
- Installed
  - Templates
    - Visual Basic
    - Visual C#
      - Store Apps
      - Windows Desktop
      - Web
        - Visual Studio 2012
      - Office/SharePoint
      - Cloud
      - LightSwitch
      - Reporting
      - Silverlight
      - Test
      - WCF
      - Workflow
    - Visual C++
    - Visual F#
    - SQL Server
    - PowerShell
- Online

.NET Framework 4.5.1 | Sort by: Default

ASP.NET Web Application — Visual C#

Search Installed Templates (Ctrl+E)

Type: Visual C#

A project template for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

□ Add Application Insights to Project

Microsoft recommends adding Application Insights telemetry to help you understand and optimize application performance.

Learn more

Privacy Statement

Click here to go online and find templates.

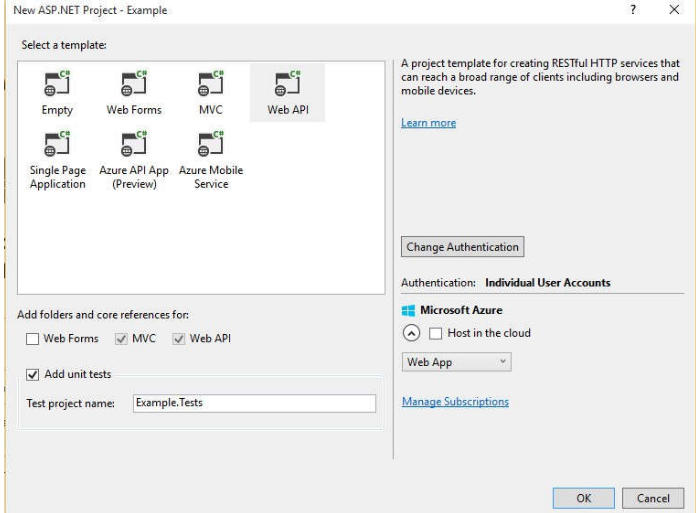
Name: ExampleOne

Location: E:\CT\Awase-CSharp\ASP.NET WEBAPI\

Solution name: ExampleOne

☑ Create directory for solution

□ Add to source control



New ASP.NET Project - Example

Select a template:

Empty, Web Forms, MVC, Web API, Single Page Application, Azure API App (Preview), Azure Mobile Service

A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.

Learn more

Authentication: **Individual User Accounts**

Microsoft Azure

□ Host in the cloud

Web App

Manage Subscriptions

Add folders and core references for:

□ Web Forms ☑ MVC ☑ Web API

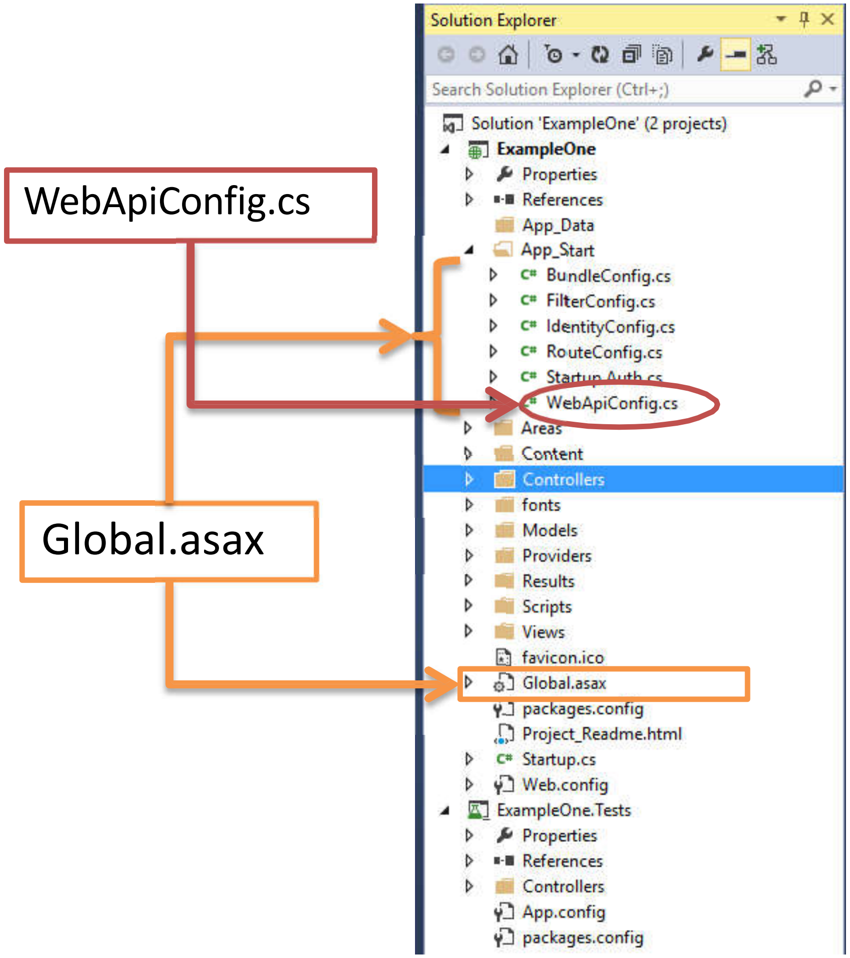
☑ Add unit tests

Test project name: Example.Tests

- Create a WebAPI Project
- Create an ASP.NET Project and add a Web API Project
- Create any project!
  - Install-package **Microsoft.AspNet.WebApi.SelfHost**
  - **Using NuGet**

C#

WEB API SCAFFOLDING PROJECT



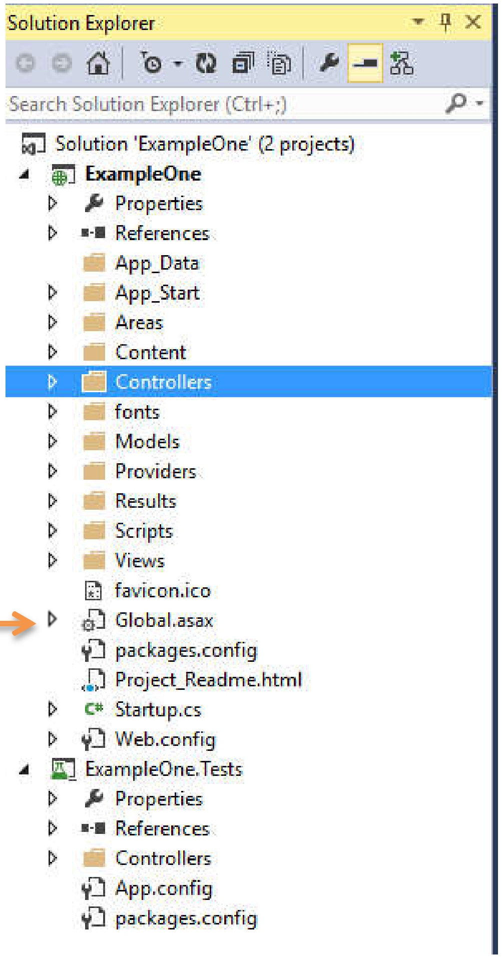
WebApiConfig.cs

Global.asax

```
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'ExampleOne' (2 projects)
  ExampleOne
    Properties
    References
    App_Data
    App_Start
      BundleConfig.cs
      FilterConfig.cs
      IdentityConfig.cs
      RouteConfig.cs
      Startup.Auth.cs
      WebApiConfig.cs
    Areas
    Content
    Controllers
    fonts
    Models
    Providers
    Results
    Scripts
    Views
    favicon.ico
    Global.asax
    packages.config
    Project_Readme.html
    Startup.cs
    Web.config
  ExampleOne.Tests
    Properties
    References
    Controllers
    App.config
    packages.config
```

WebAPI ASP.NET MVC5 Project

Global.asax



```
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'ExampleOne' (2 projects)
  ExampleOne
    Properties
    References
    App_Data
    App_Start
    Areas
    Content
    Controllers
    fonts
    Models
    Providers
    Results
    Scripts
    Views
    favicon.ico
    Global.asax
    packages.config
    Project_Readme.html
    Startup.cs
    Web.config
  ExampleOne.Tests
    Properties
    References
    Controllers
    App.config
    packages.config
```

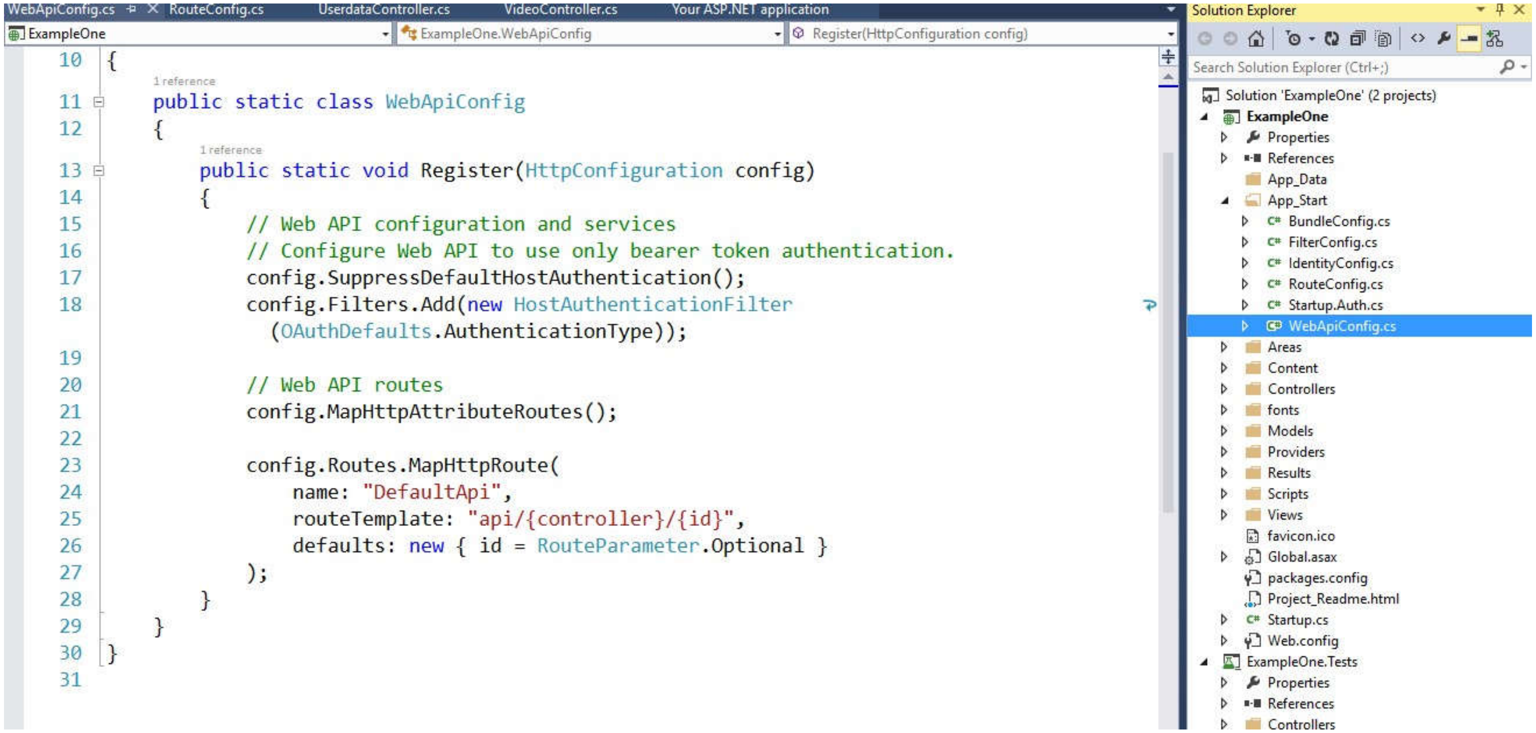
Conventional Routing

- Convention-based routing, you define one or more route templates, which are basically parameterized strings.
- ASP.NET Hosting Layer Maps URIs and Verbs to controller
  - Must derive from **APIController**
- **New Extension Method for Routing**
  - **MapHttpRoute**
  - **Registers a differentHandler to the routing Infrastructure**
    - **HttpControllerHandler**

- Advantage being that templates are defined in a single place and routing rules are applied consistently across controllers
- ASP.NET Routing is the most common way to map URIs and Verbs to your methods
  - Not the only way however
  - Self Hosting can use this system or replace it with your own.

C#

Conventional Routing

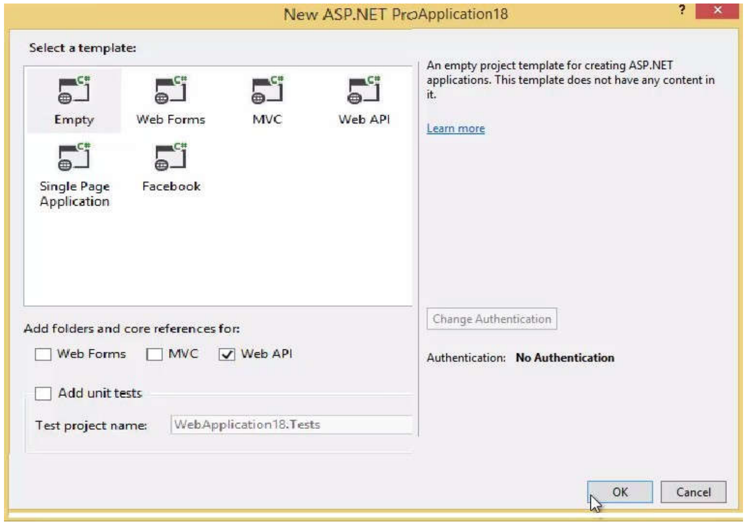


```
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            // Web API configuration and services
            // Configure Web API to use only bearer token authentication.
            config.SuppressDefaultHostAuthentication();
            config.Filters.Add(new HostAuthenticationFilter
              (OAuthDefaults.AuthenticationType));

            // Web API routes
            config.MapHttpAttributeRoutes();

            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
        }
    }
}
```

```
// Web API routes
config.MapHttpAttributeRoutes();

config.Routes.MapHttpRoute(
    name: "DefaultApi",
    routeTemplate: "api/{controller}/{id}",
    defaults: new { id = RouteParameter.Optional }
);
```

C#

Conventional Routing



New ASP.NET ProoApplication18

Select a template:

An empty project template for creating ASP.NET applications. This template does not have any content in it.

Learn more

Add folders and core references for:

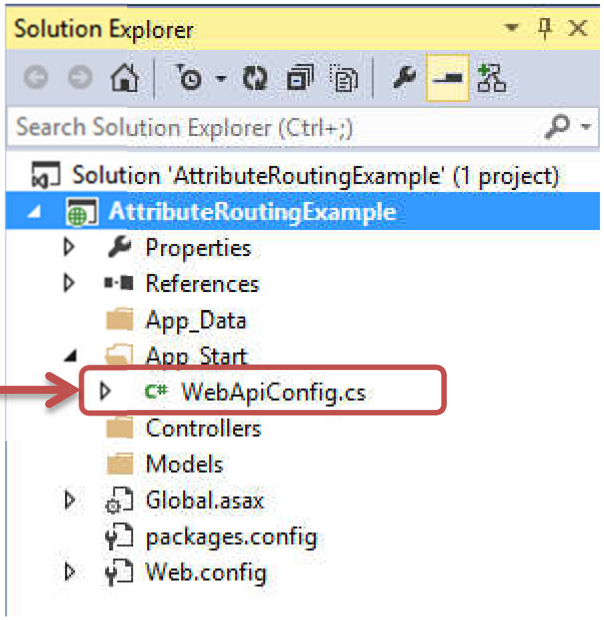
□ Web Forms □ MVC ☑ Web API

□ Add unit tests

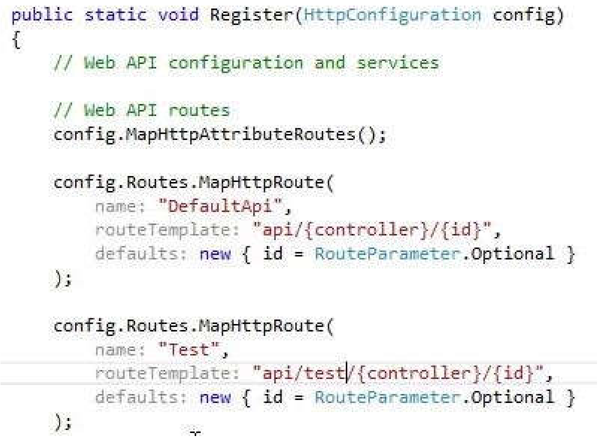
Test project name: WebApplication18.Tests

Authentication: **No Authentication**

WebAPI Configuration Services
And WebAPI Routes



Solution 'AttributeRoutingExample' (1 project)
- AttributeRoutingExample
  - Properties
  - References
  - App_Data
  - App_Start
    - WebApiConfig.cs
  - Controllers
  - Models
  - Global.asax
  - packages.config
  - Web.config

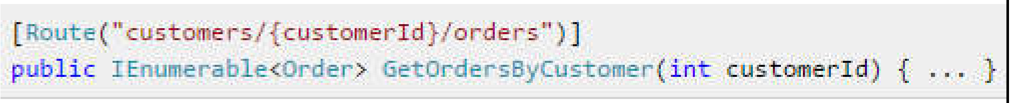


```
public static void Register(HttpConfiguration config)
{
    // Web API configuration and services

    // Web API routes
    config.MapHttpAttributeRoutes();

    config.Routes.MapHttpRoute(
        name: "DefaultApi",
        routeTemplate: "api/{controller}/{id}",
        defaults: new { id = RouteParameter.Optional }
    );

    config.Routes.MapHttpRoute(
        name: "Test",
        routeTemplate: "api/test/{controller}/{id}",
        defaults: new { id = RouteParameter.Optional }
    );
```

Attribute Routing

- Convention-based routing makes it hard to support certain URI patterns that are common in RESTful APIs.
- E.g. Customers have orders, movies have actors, books have authors
- It’s natural to create URIs that reflect these relations
  - **/customers/1/orders**

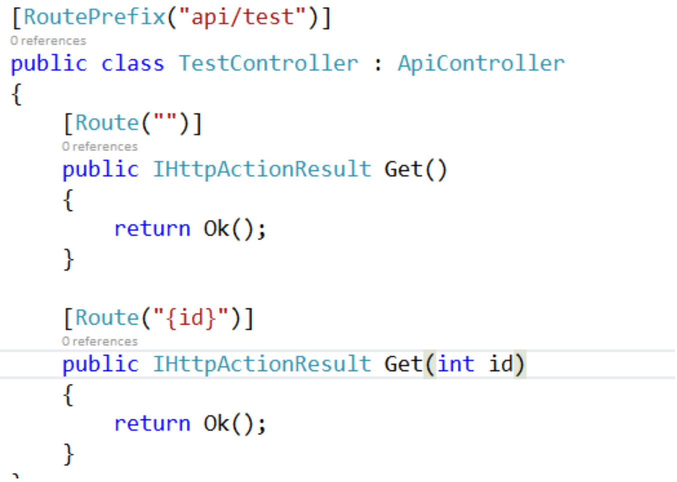
- Attribute routing, is trivial to define a route for this URL.
- Simply by adding an [attribute] to the controller action



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

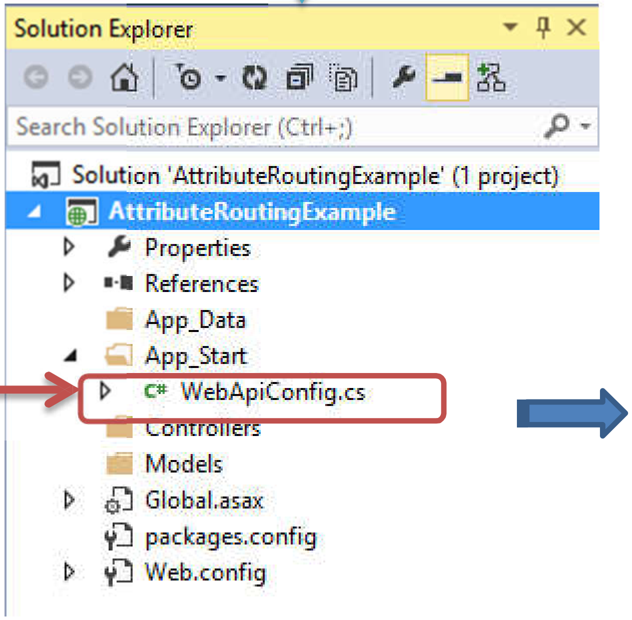
Attribute routing

set a common prefix for an entire controller by using the **[RoutePrefix]** attribute:

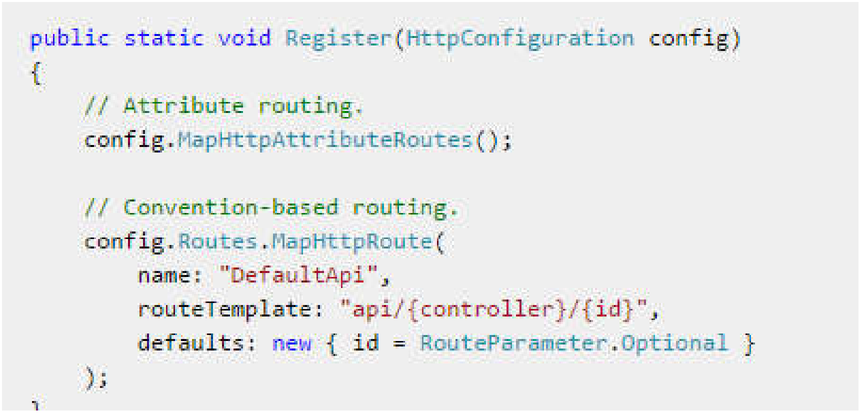


```
[RoutePrefix("api/test")]
0 references
public class TestController : ApiController
{
    [Route("")]
    0 references
    public IHttpActionResult Get()
    {
        return Ok();
    }

    [Route("{id}")]
    0 references
    public IHttpActionResult Get(int id)
    {
        return Ok();
    }
}
```

WebAPI Configuration Services And WebAPI Routes



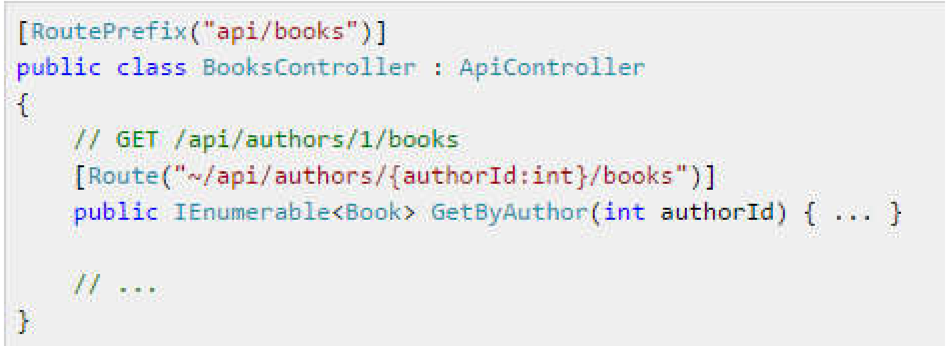
Enabling Attribute Routing



```
public static void Register(HttpConfiguration config)
{
    // Attribute routing.
    config.MapHttpAttributeRoutes();

    // Convention-based routing.
    config.Routes.MapHttpRoute(
        name: "DefaultApi",
        routeTemplate: "api/{controller}/{id}",
        defaults: new { id = RouteParameter.Optional }
    );
}
```

Override the route prefix

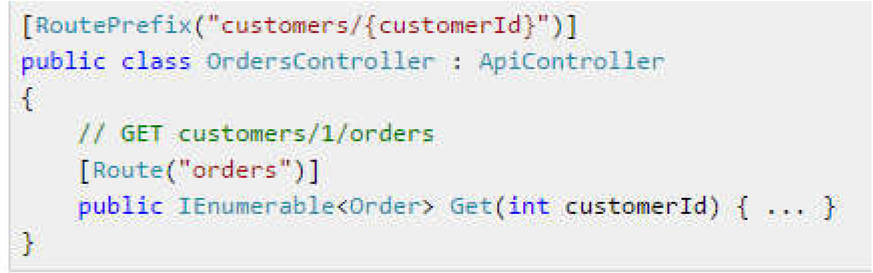
Use a **tilde (~)** on the method attribute to override the route prefix:



```
[RoutePrefix("api/books")]
public class BooksController : ApiController
{
    // GET /api/authors/1/books
    [Route("~/api/authors/{authorId:int}/books")]
    public IEnumerable<Book> GetByAuthor(int authorId) { ... }

    // ...
}
```

Route prefix parameters

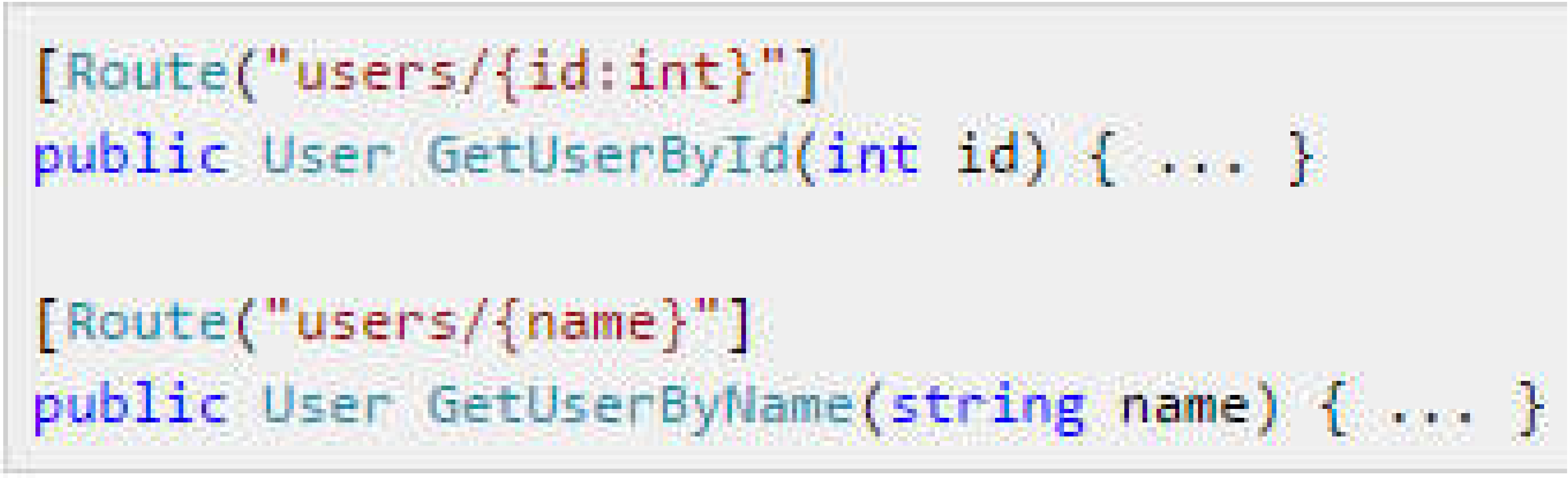
route prefix can include parameters:



```
[RoutePrefix("customers/{customerId}")]
public class OrdersController : ApiController
{
    // GET customers/1/orders
    [Route("orders")]
    public IEnumerable<Order> Get(int customerId) { ... }
}
```

Route constraints

Route constraints let you restrict how the parameters in the route template are matched. The general syntax is "**{parameter:constraint}"**



```
[Route("users/{id:int}"]
public User GetUserById(int id) { ... }

[Route("users/{name}"]
public User GetUserByName(string name) { ... }
```

Supported constraints

Constraint	Description	Example
alpha	Matches uppercase or lowercase Latin alphabet characters (a-z, A-Z)	{x:alpha}
bool	Matches a Boolean value.	{x:bool}
datetime	Matches a **DateTime** value.	{x:datetime}
decimal	Matches a decimal value.	{x:decimal}
double	Matches a 64-bit floating-point value.	{x:double}
float	Matches a 32-bit floating-point value.	{x:float}
guid	Matches a GUID value.	{x:guid}

C#

int	Matches a 32-bit integer value.	{x:int}

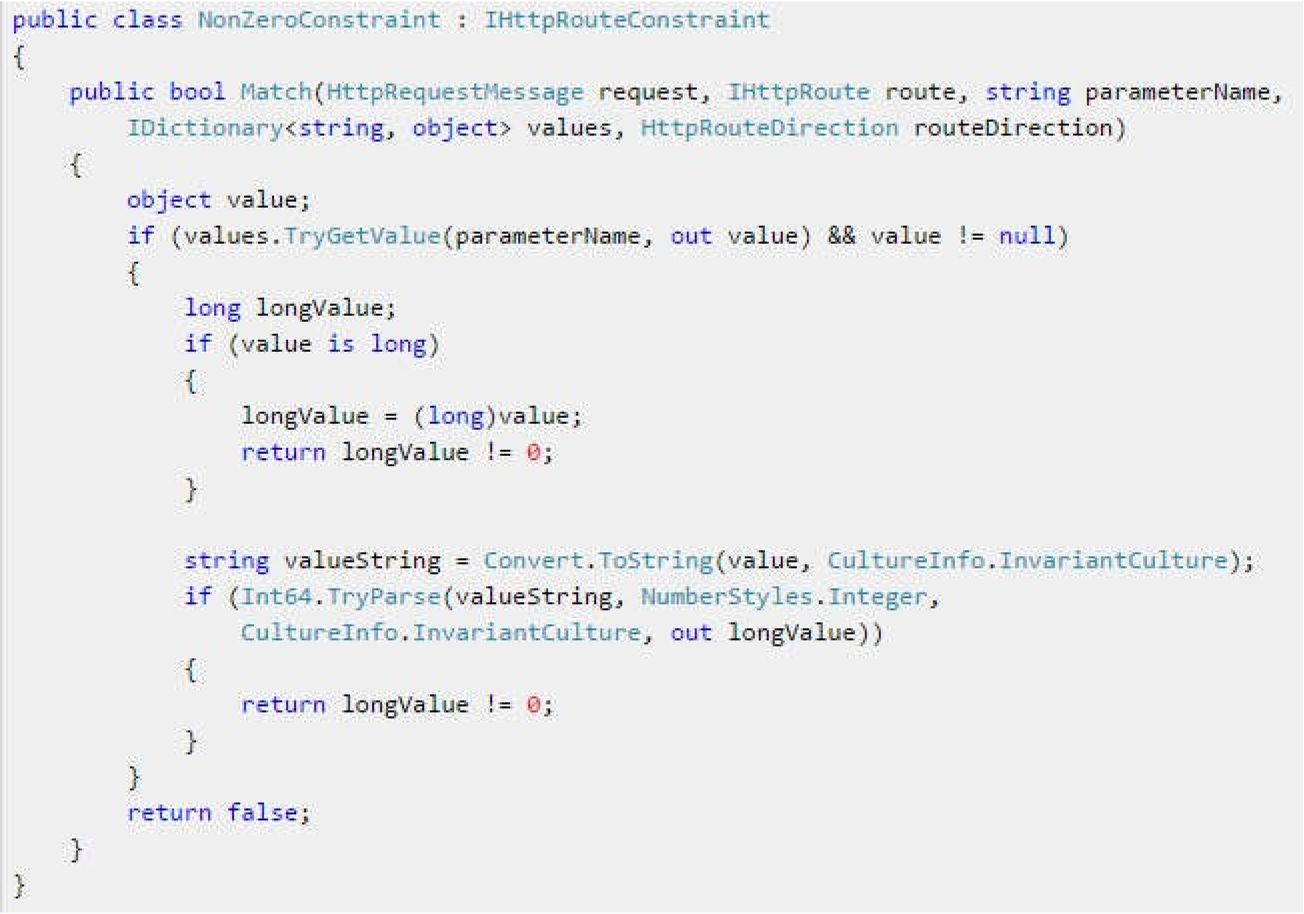
length	Matches a string with the specified length or within a specified range of lengths.	{x:length(6)}
		{x:length(1,20)}

long	Matches a 64-bit integer value.	{x:long}
max	Matches an integer with a maximum value.	{x:max(10)}
maxlength	Matches a string with a maximum length.	{x:maxlength(10)}
min	Matches an integer with a minimum value.	{x:min(10)}
minlength	Matches a string with a minimum length.	{x:minlength(10)}
range	Matches an integer within a range of values.	{x:range(10,50)}
regex	Matches a regular expression.	{x:regex(^\d{3}-\d{3}-\d{4}$)}

Custom Route Constraints

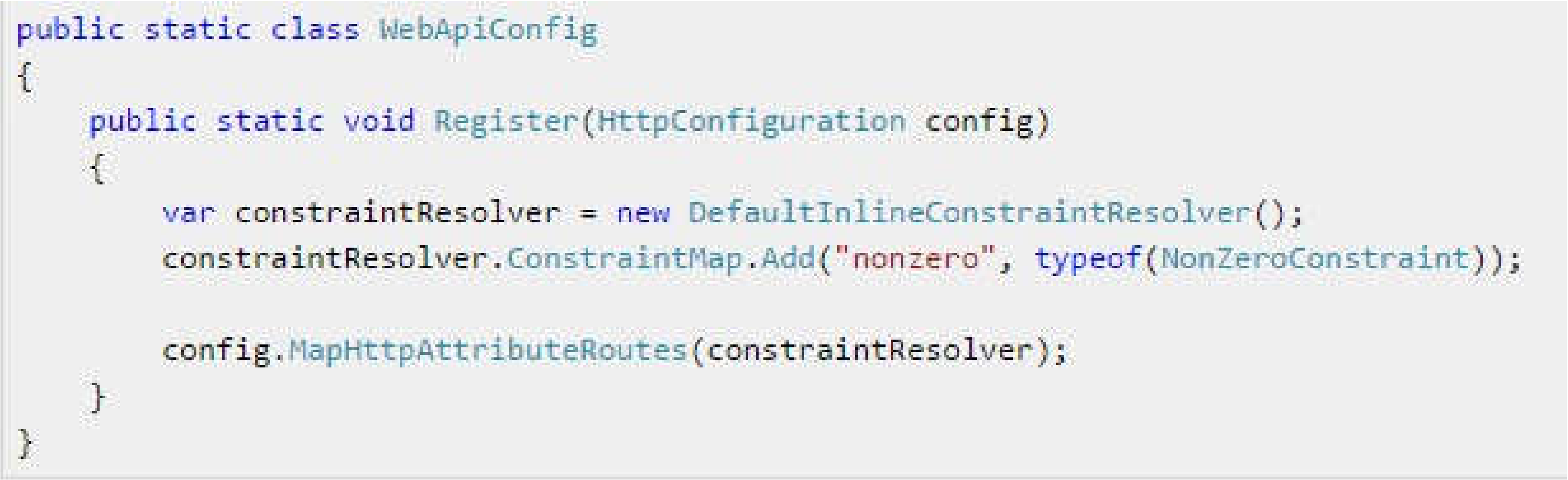
custom route constraints by implementing the **IHttpRouteConstraint** interface.

constraint restricts a parameter to a non-zero integer value.

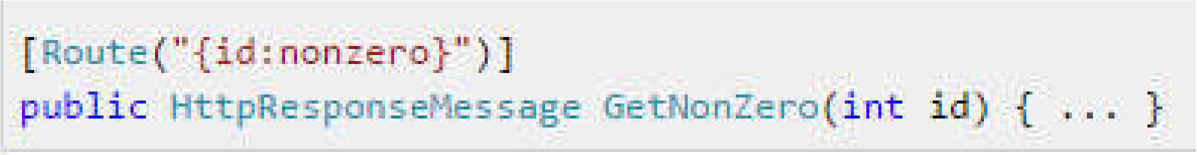


```
public class NonZeroConstraint : IHttpRouteConstraint
{
    public bool Match(HttpRequestMessage request, IHttpRoute route, string parameterName,
        IDictionary<string, object> values, HttpRouteDirection routeDirection)
    {
        object value;
        if (values.TryGetValue(parameterName, out value) && value != null)
        {
            long longValue;
            if (value is long)
            {
                longValue = (long)value;
                return longValue != 0;
            }

            string valueString = Convert.ToString(value, CultureInfo.InvariantCulture);
            if (Int64.TryParse(valueString, NumberStyles.Integer,
                CultureInfo.InvariantCulture, out longValue))
            {
                return longValue != 0;
            }
        }
        return false;
    }
}
```

Custom Route Constraint



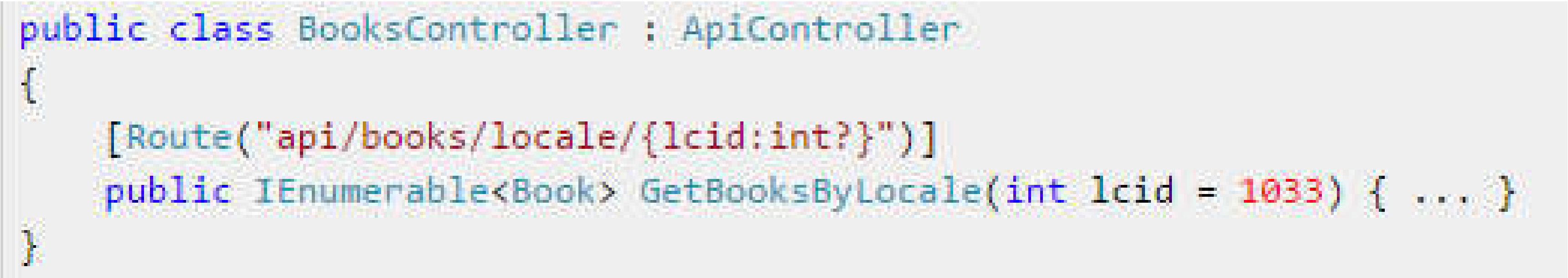
```
public static class WebApiConfig
{
    public static void Register(HttpConfiguration config)
    {
        var constraintResolver = new DefaultInlineConstraintResolver();
        constraintResolver.ConstraintMap.Add("nonzero", typeof(NonZeroConstraint));

        config.MapHttpAttributeRoutes(constraintResolver);
    }
}
```



```
[Route("{id:nonzero}")]
public HttpResponseMessage GetNonZero(int id) { ... }
```

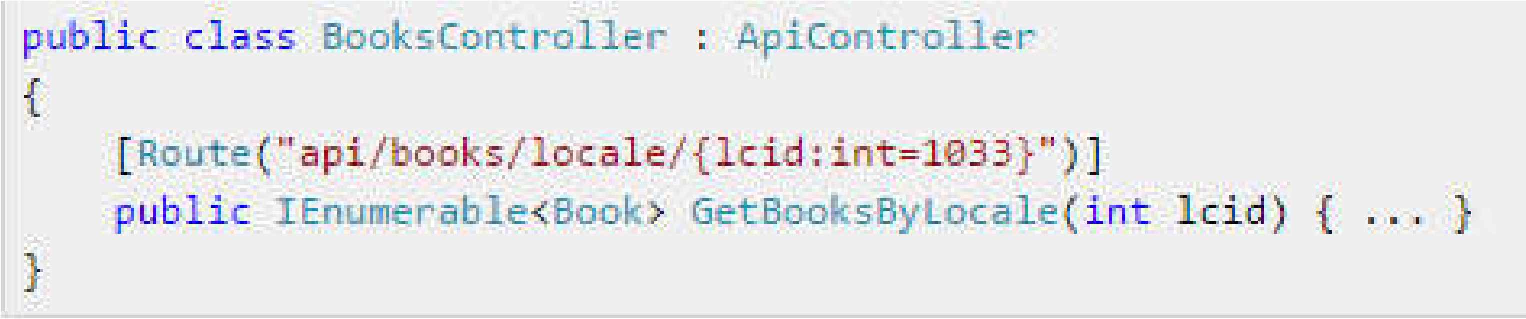
Optional URI Parameters and Default Values

make a URI parameter optional by adding a question mark to the route parameter. If a route parameter is optional, you must define a default value for the method parameter.



```
public class BooksController : ApiController
{
    [Route("api/books/locale/{lcid:int?}")]
    public IEnumerable<Book> GetBooksByLocale(int lcid = 1033) { ... }
}
```

[OR]

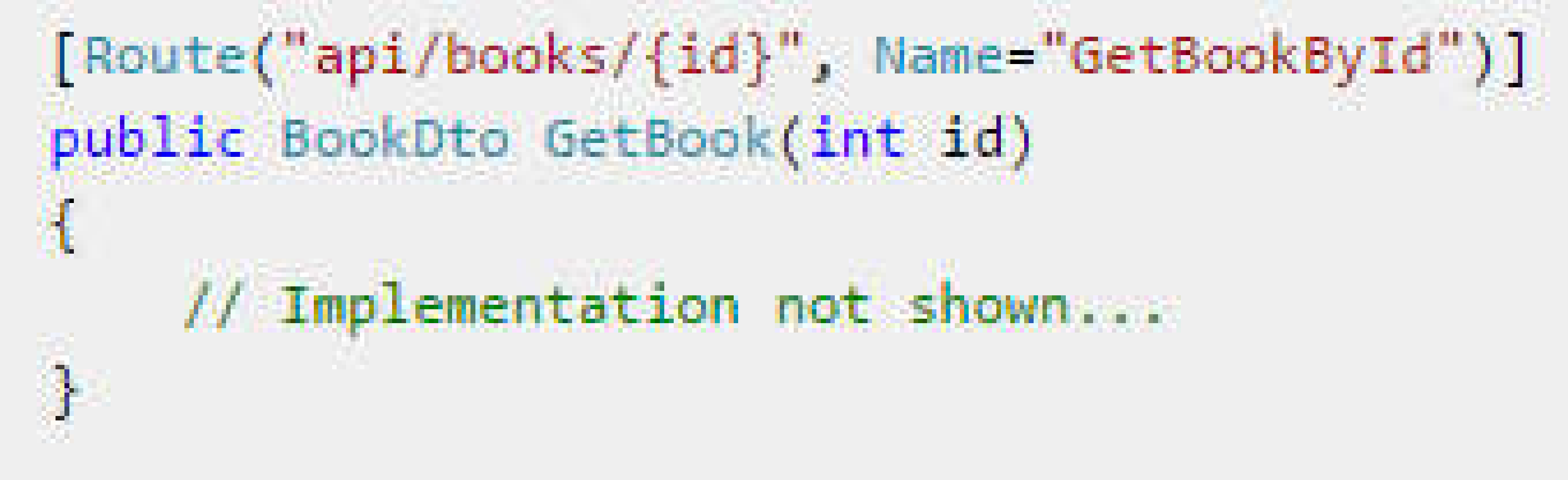


```
public class BooksController : ApiController
{
    [Route("api/books/locale/{lcid:int=1033}")]
    public IEnumerable<Book> GetBooksByLocale(int lcid) { ... }
}
```

Route Name

In Web API, every route has a name. Route names are useful for generating links, so that you can include a link in an HTTP response.

set the **Name** property on the attribute.



```
[Route("api/books/{id}", Name="GetBookById")]
public BookDto GetBook(int id)
{
    // Implementation not shown...
}
```

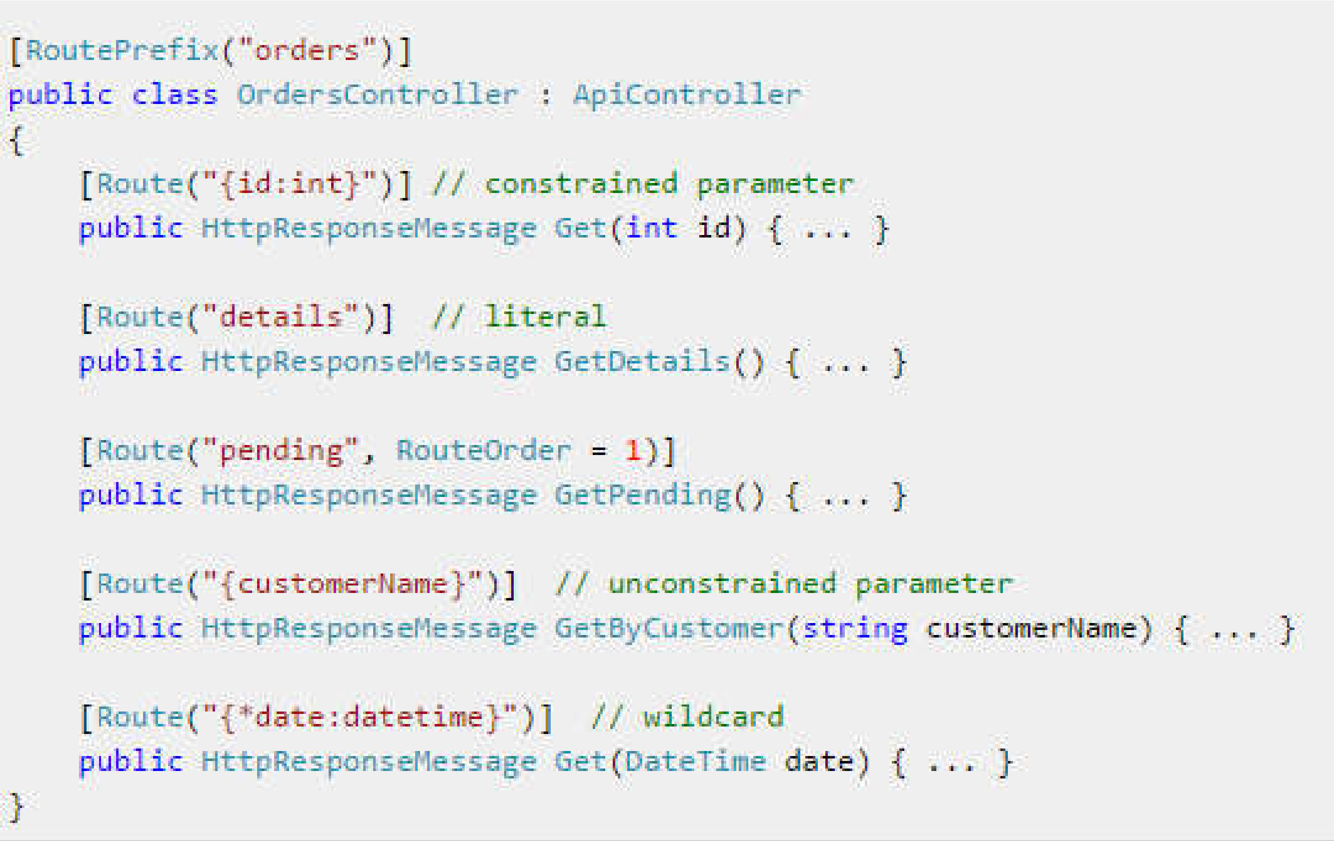
Route Order

When the framework tries to match a URI with a route, it evaluates the routes in a particular order. To specify the order, set the **RouteOrder** property on the route attribute. Lower values are evaluated first. The default order value is zero.

How is the total ordering is determined ?

1. Compare the **RouteOrder** property of the route attribute.
2. Look at each URI segment in the route template. For each segment, order as follows:
   i. Literal segments.
   ii. Route parameters with constraints.
   iii. Route parameters without constraints.
   iv. Wildcard parameter segments with constraints.
   v. Wildcard parameter segments without constraints.
3. In the case of a tie, routes are ordered by a case-insensitive ordinal string comparison (OrdinalIgnoreCase) of the route template.

Route Order



```
[RoutePrefix("orders")]
public class OrdersController : ApiController
{
    [Route("{id:int}")] // constrained parameter
    public HttpResponseMessage Get(int id) { ... }

    [Route("details")]  // literal
    public HttpResponseMessage GetDetails() { ... }

    [Route("pending", RouteOrder = 1)]
    public HttpResponseMessage GetPending() { ... }

    [Route("{customerName}")]  // unconstrained parameter
    public HttpResponseMessage GetByCustomer(string customerName) { ... }

    [Route("{*date:datetime}")]  // wildcard
    public HttpResponseMessage Get(DateTime date) { ... }
}
```

1. orders/details
2. orders/{id}
3. orders/{customerName}
4. orders/{*date}
5. orders/pending

Conventional Routing

- Define one or more templates, which are basically parameterized strings.
- Templates are defined in a single place and routing rules are applied consistently across all controllers

Attribute Routing

- Convention based routing makes it hard to support certain URI patterns that are common in RESTful APIs.
- Applied to Controller Action Methods or at the Controller level.
- Ability to define constraints and custom route constraints
- Route names are useful for generating links, so that they can include a link in an HTTP response

SUMMARY

- Approaches for creating web api 2 applications
- Conventional routing using **MapHttpRoute**
- **Attribute Routing approaches**
- **In fact, you can combine both techniques in the same project.**

SYED AWASE KHIRNI

ACTION RESULTS IN WEB API

ACTION RESULTS

- Converting the return value from a controller action into an HTTP response message.
- WEB API 1 : two ways of creating response from an API action
  - Either return a specific object instance (or void) and let the Web API pipeline convert that to an **HttpResponseMessage** for you
  - Return a raw HttpResponseMessage, where the user has to construct it manually and bypass all of the internal Web API mechanisms (formatters, content negotiation).
- WEB API 2: **IHTTPACTIONRESULT is a kind of wrap of HTTPRESPONSEMESSAGE. It contains ExecuteAsync method to create an HttpResponseMessage, further simplifies unit testing of your controller. The return type are kind of strongly typed classes serialized by Web API using media formatter into the response body.**
  - **Drawbacks being one cannot directly return an error code, such as 404. Instead you can only throw and HttpResponseException error.**

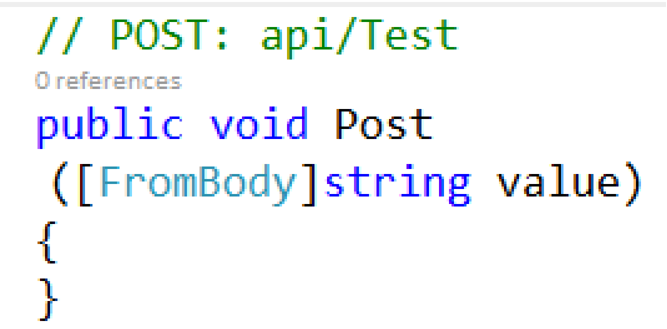
WEB API Controller Action

- Web API uses different mechanisms to create the **HTTP response**

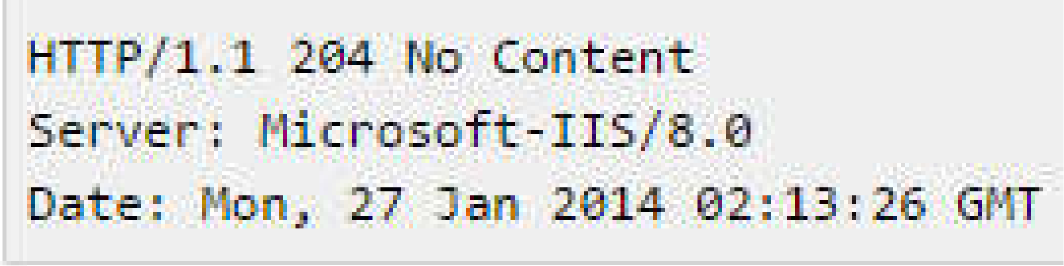
Return type	How Web API create the response
Void	Return empty 204 (No Content)
HttpResponseMessage	Convert directly to an HTTP response message
IHTTPActionResult	Call ExecuteAsync to create an HTTPResponseMessage, then convert to an HTTP response message
Other type	Write the serialized return value into the response body; return 200(OK)

VOID

- If the return type is void, Web API simply returns an empty HTTP response with status code **204 (NO CONTENT)**



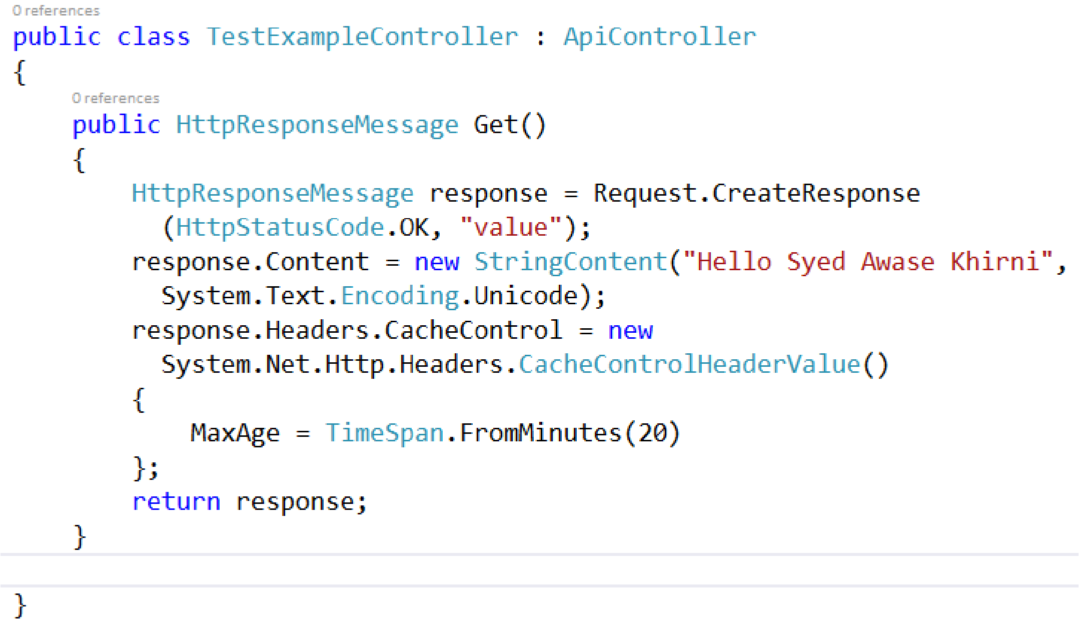
```
// POST: api/Test
0 references
public void Post
 ([FromBody]string value)
{
}
```



```
HTTP/1.1 204 No Content
Server: Microsoft-IIS/8.0
Date: Mon, 27 Jan 2014 02:13:26 GMT
```

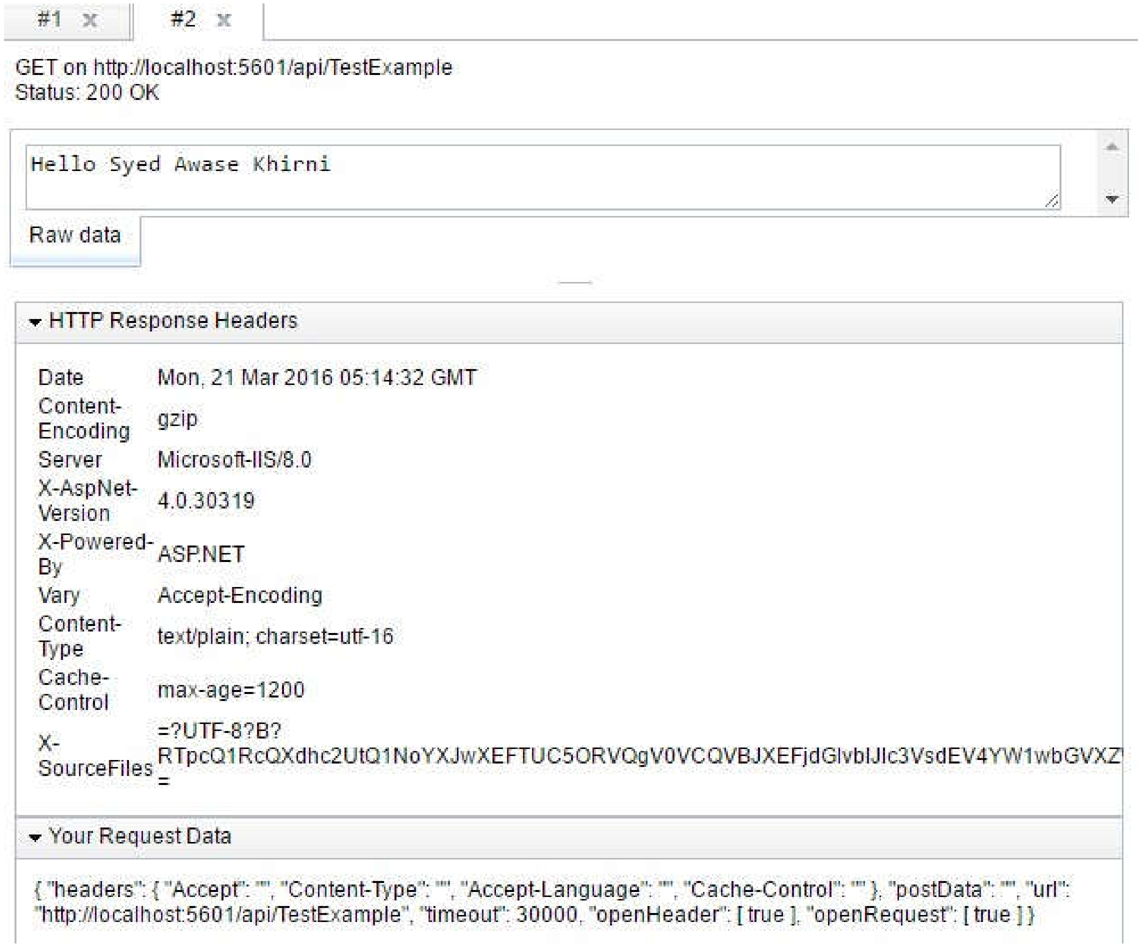
HttpResponseMessage

- **HttpResponseMessage** return type, converts the return value directly into an HTTP response message using the properties of the **HttpResponseMessage** object to populate the response.
- The option gives you a lot of control over the response message



```
0 references
public class TestExampleController : ApiController
{
    0 references
    public HttpResponseMessage Get()
    {
        HttpResponseMessage response = Request.CreateResponse
          (HttpStatusCode.OK, "value");
        response.Content = new StringContent("Hello Syed Awase Khirni",
          System.Text.Encoding.Unicode);
        response.Headers.CacheControl = new
          System.Net.Http.Headers.CacheControlHeaderValue()
        {
            MaxAge = TimeSpan.FromMinutes(20)
        };
        return response;
    }

}
```

HttpResponseMessage



GET on http://localhost:5601/api/TestExample
Status: 200 OK

```
Hello Syed Awase Khirni
```

HTTP Response Headers

Date	Mon, 21 Mar 2016 05:14:32 GMT
Content-Encoding	gzip
Server	Microsoft-IIS/8.0
X-AspNet-Version	4.0.30319
X-Powered-By	ASP.NET
Vary	Accept-Encoding
Content-Type	text/plain; charset=utf-16
Cache-Control	max-age=1200
X-SourceFiles	=?UTF-8?B? RTpcQ1RcQXdhc2UtQ1NoYXJwXEFTUC5ORVQgV0VCQVBJXEFjdGlvbIJlc3VsdEV4YW1wbGVXZ =

Your Request Data

{ "headers": { "Accept": "", "Content-Type": "", "Accept-Language": "", "Cache-Control": "" }, "postData": "", "url": "http://localhost:5601/api/TestExample", "timeout": 30000, "openHeader": [true], "openRequest": [true] }

Content Negotiation in ASP.NET WEB API

"the process of selecting the best representation for a given response when there are multiple representations available"

RFC 2616

Primary Mechanism for Content Negotiation in HTTP are using these request headers

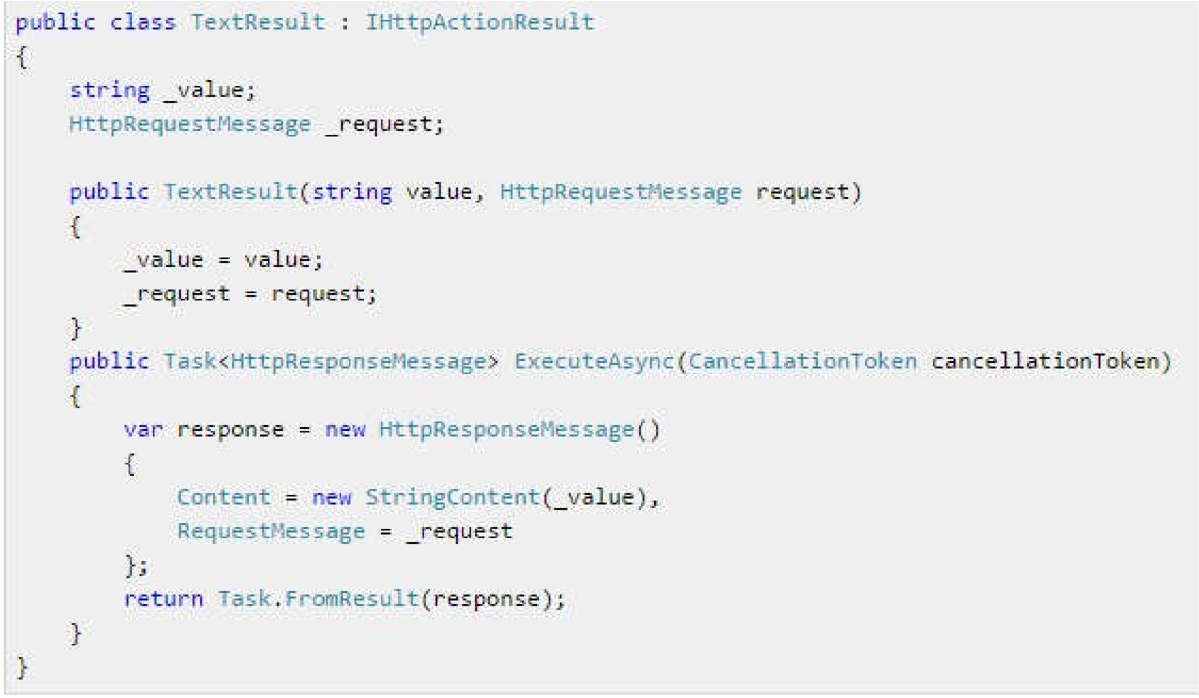
1. **Accept :** which media types are acceptable for the response, such as “application/json”, “application/xml” or a custom media type such as “application/vnc.example+xml”
2. **Accept-Charset**: which character sets are acceptable such as UTF-8 or ISO 8859-1
3. **Accept-Encoding:** which content encoding are acceptable, such as gzip.
4. **Accept-Language:** preferred natural language, such as “en-us”
5. **Other Option – if the request contains an X-Requested-With header, indicating an AJAX request, the server might default to JSON if there is no Accept header**

IHttpActionResult

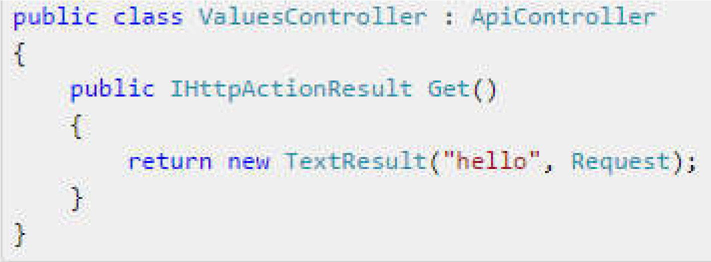
- Introduced in Web API2
- IHttpActionResult Interface defines an HttpResponseMessage factory
- Advantages of using IHttpActionResult
  - Simplifies unit testing webapi controllers
  - Moves common logic for creating HTTP responses into separate classes
  - Makes the intent of the controller action clearer, by hiding the low-level details of constructing the response

IHttpActionResult

Contains a single method, **ExecuteAsync,** which asynchronously creates an **HttpResponseMessage** instance



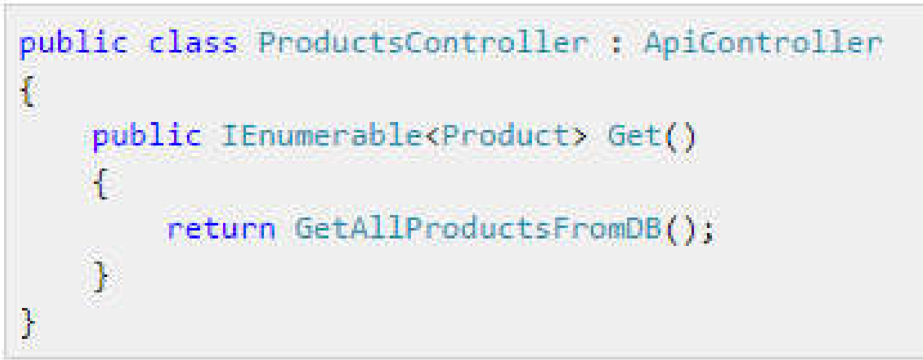
```
public class TextResult : IHttpActionResult
{
    string _value;
    HttpRequestMessage _request;

    public TextResult(string value, HttpRequestMessage request)
    {
        _value = value;
        _request = request;
    }
    public Task<HttpResponseMessage> ExecuteAsync(CancellationToken cancellationToken)
    {
        var response = new HttpResponseMessage()
        {
            Content = new StringContent(_value),
            RequestMessage = _request
        };
        return Task.FromResult(response);
    }
}
```



```
public class ValuesController : ApiController
{
    public IHttpActionResult Get()
    {
        return new TextResult("hello", Request);
    }
}
```

Other Return Types

- Web API uses a media formatter to serialize the return value. Web API writes the serialized value into the response body.
- A disadvantage of this approach is that you cannot directly return an error code e.g. 404.
- You can only throw an **HttpResponseException** for error codes.



```
public class ProductsController : ApiController
{
    public IEnumerable<Product> Get()
    {
        return GetAllProductsFromDB();
    }
}
```

Assemblies

System.Net.Http	• Client and raw messaging types
System.Net.Http.Formatting	• Model Binding and media type formatter
System.Web.Http	• Basic Hosting Infrastructure
System.Web.Http.Common	• Common APIs
System.Web.Http.WebHost	• ASP.NET hosting
System.Web.Http.SelfHost	• Self hosting
System.Web.Http.Data	• DataController is an APIController that handles “CRUD” type operations
System.Web.Http.Data.EntityFramework	• Specific implementations of DataController
System.Web.Http.Data.Helpers	• Common code for data api

UNIFORM INTERFACE

- The Uniform interface of REST
- ASP.NET Web API and the uniform interface
- Content Negotiation
- Model binding and formatting
- HTTP-deep diving

REST

- Representational State Transfer
- Architecture for building systems (by Roy Fielding)
- Based on the advantages of the Web
  - URIs
  - Uniform Interface
  - Stateless
  - Hypermedia-driven(i.e links)
  - Cache-ability

It’s all about URIs

- A RESTful service models its resources as URIs
  - Builds on the success of the web
- Everything is addressable via a URI
- You interact with a resource by using the Uniform Interface
  - Start with URI, add well-known HTTP verbs

Uniform interface

GET

- No Side-effects (Safe)
- Idempotent(calling a million times has the same effect as one request)
- Retrieves resource
- Cacheable

POST

- Creates a new resource
- Same as SOAP – still unsafe

PUT

- Updates an existing resource
- Also idempotent

DELETE

- Removes a resource
- Also idempotent

Implementing using Conventions

- The route determines which Controller should be invoked based on the request URI
- Controller derives from APIController
- Method invoked is picked based upon the verb of the incoming HTTP request
  - GET
  - POST
  - PUT
  - DELETE

Verbs to Attributes

- If you don’t want to name your methods with the Convention, you can add attributes to your methods
  - Name method whatever you’d like
  - Routing still picks Controller
- AcceptVerbs
  - Can specify multiple verbs to one method
- Specific attributes
  - HttpGet
  - HttpPost
  - HttpPut
  - HttpDelete

Content Negotiation

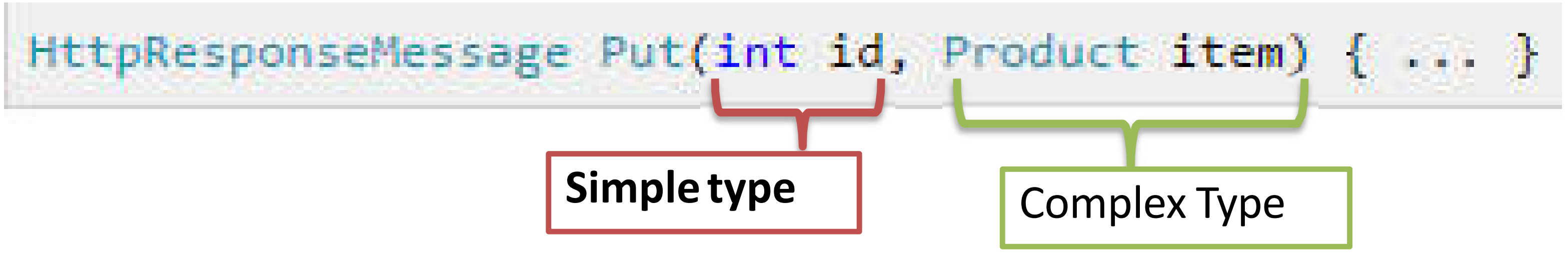
- **Dynamically determining the media type of a resource based on client request**
  - Client sends Accept header with 1…N media types (XML/JSON)
  - Server sends back appropriate response with Content-Type header
  - Client also sends Content-Type header when sending a body
- Web API provides automatic content negotiation
- Implemented using **MediaTypeFormatter** base class
  - More on this in extensibility Module

Parameter Binding in ASP.NET WEB API

- When a WEB API calls a method on a controller, it must set values for the parameters, a process called binding.

Rules of binding

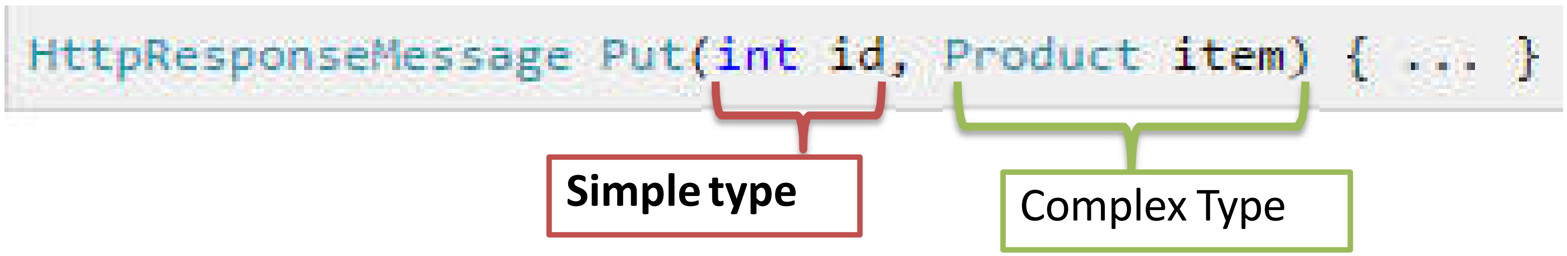
- **Simple types** – include .NET primitive types (int, bool, double ..+ TimeSpan, DateTime, Guid, decimal and String)
- **Complex types** –Web API tries to read the value from the message body, using a media-type formatter



```
HttpResponseMessage Put(int id, Product item) { ... }
```

Simple type

Complex Type



```
HttpResponseMessage Put(int id, Product item) { ... }
```

Simple type

Complex Type

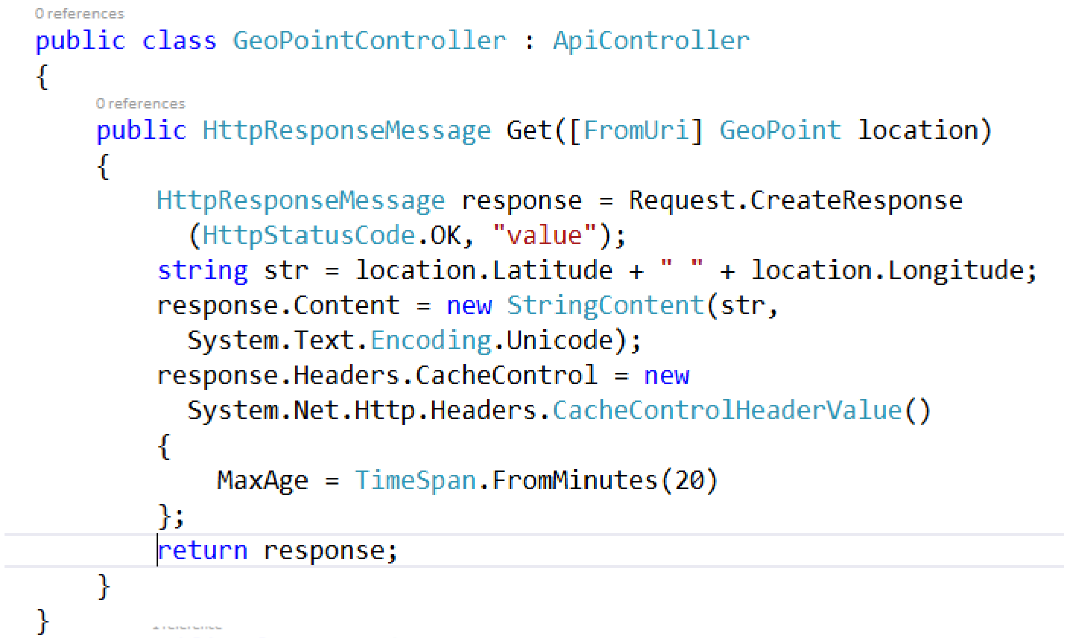
id – parameter is of **simple type and gets the value from the request URI**

Item-parameter is a **complex type**, so WEB API uses a media-type formatter to read the value from the request body

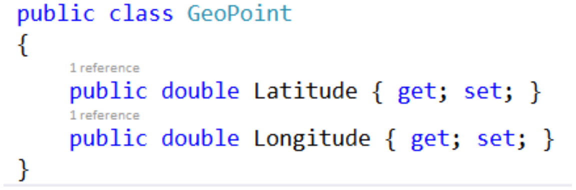
For complex types, however, consider using **media-type formatters whenever possible.**<u>A key principle of HTTP is that resources are sent in the message body, using content negotiation to specify the representation of the resource. Media-type formatters were designed for exactly this purpose.</u>

[FromURI]

- To force WEB API to read a complex type from the URI
- Add the **[FromUri]** attribute to the parameter.



```
0 references
public class GeoPointController : ApiController
{
    0 references
    public HttpResponseMessage Get([FromUri] GeoPoint location)
    {
        HttpResponseMessage response = Request.CreateResponse
          (HttpStatusCode.OK, "value");
        string str = location.Latitude + " " + location.Longitude;
        response.Content = new StringContent(str,
          System.Text.Encoding.Unicode);
        response.Headers.CacheControl = new
          System.Net.Http.Headers.CacheControlHeaderValue()
        {
            MaxAge = TimeSpan.FromMinutes(20)
        };
        return response;
    }
}
```



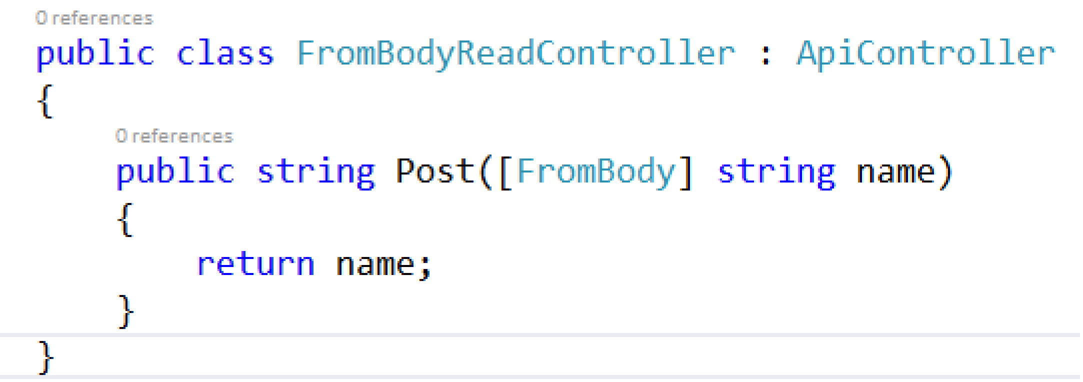
```
public class GeoPoint
{
    1 reference
    public double Latitude { get; set; }
    1 reference
    public double Longitude { get; set; }
}
```

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

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

[FromBody]

- To force WEB API to read a **simple type from the request body, add the [FromBody] attribute to the parameter**
- **At most one parameter is allowed to read from the message body.**



```
0 references
public class FromBodyReadController : ApiController
{
    0 references
    public string Post([FromBody] string name)
    {
        return name;
    }
}
```

Content-Type: application/json

The reason for this rule is that the request body might be stored in a non-buffered stream that can only be read once.

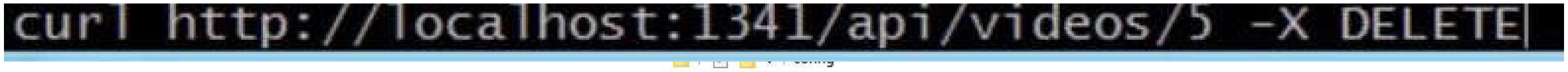
ModelBinding and Formatting

- The Web API will automatically bind URI and HTTP Body data to your methods
  - Body data can be passed to **MediaTypeFormatter** based on Content-Type
  - Model Binding (same as ASP.NET MVC) is used on Query String (URI) data
- Rule-body can only be read once
- **[FromBody]** and **[FromUri]** attributes used to control
- ModelBinding attribute can specify a custom ModelBinding for a parameter

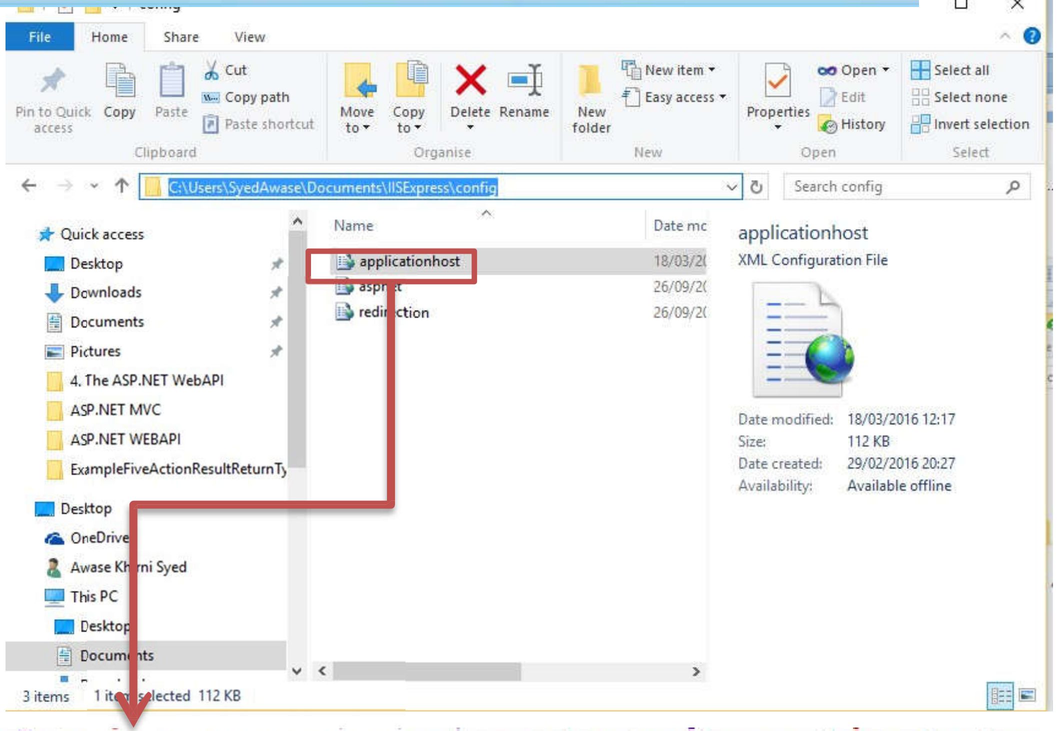
HTTP-Diving Deeper

- Routing and controllers take care of the basics of HTTP
  - URI + Verb
- What if you want to reach down further into the HTTP stack?
  - Send back particular return methods
  - Interrogate additional HTTP headers
- HttpRequestMessage/HttpResponseMessage are the answer in the Web API
  - HttpRequestMessage can replace body parameter
  - HttpResponseMessage can replace return parameter

Enable all http methods to come through

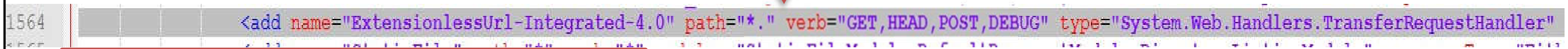


```
curl http://localhost:1341/api/videos/5 -X DELETE
```



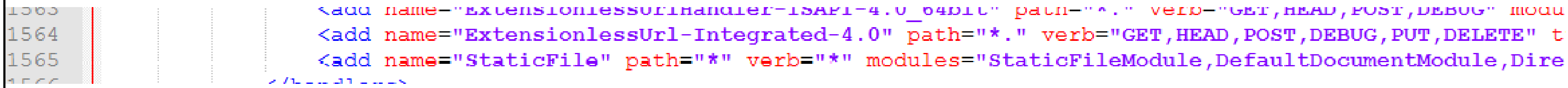
C:\Users\SyedAwase\Documents\IISExpress\config

Current http methods supported are



```
<add name="ExtensionlessUrl-Integrated-4.0" path="*." verb="GET,HEAD,POST,DEBUG" type="System.Web.Handlers.TransferRequestHandler"
```

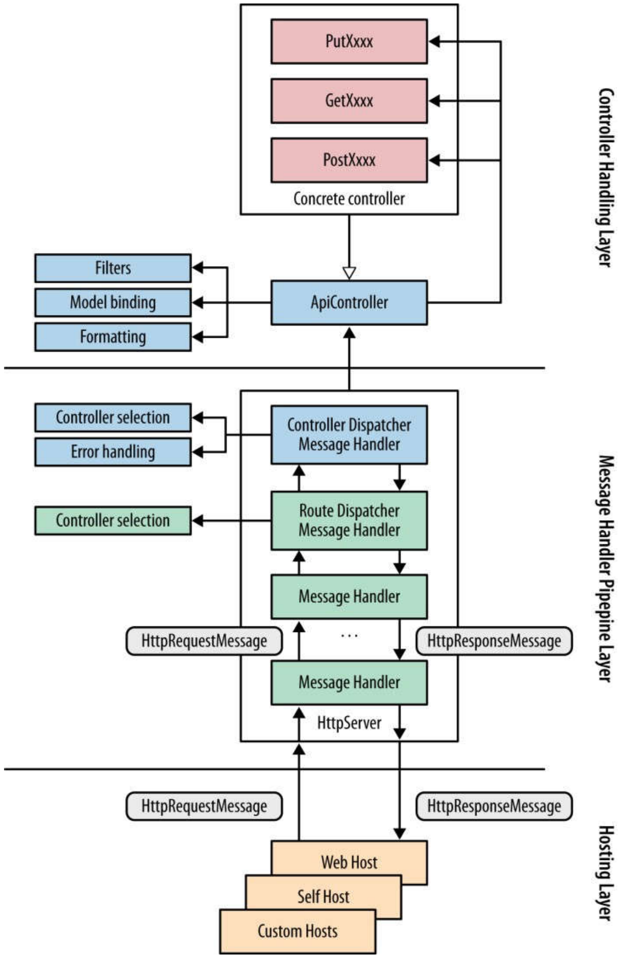
Enable for/change/ Add “Put”, “DELETE”



```
<add name="ExtensionlessUrlHandler-ISAPI-4.0_64bit" path="*." verb="GET,HEAD,POST,DEBUG" modu
<add name="ExtensionlessUrl-Integrated-4.0" path="*." verb="GET,HEAD,POST,DEBUG,PUT,DELETE" t
<add name="StaticFile" path="*" verb="*" modules="StaticFileModule,DefaultDocumentModule,Dire
```

C#

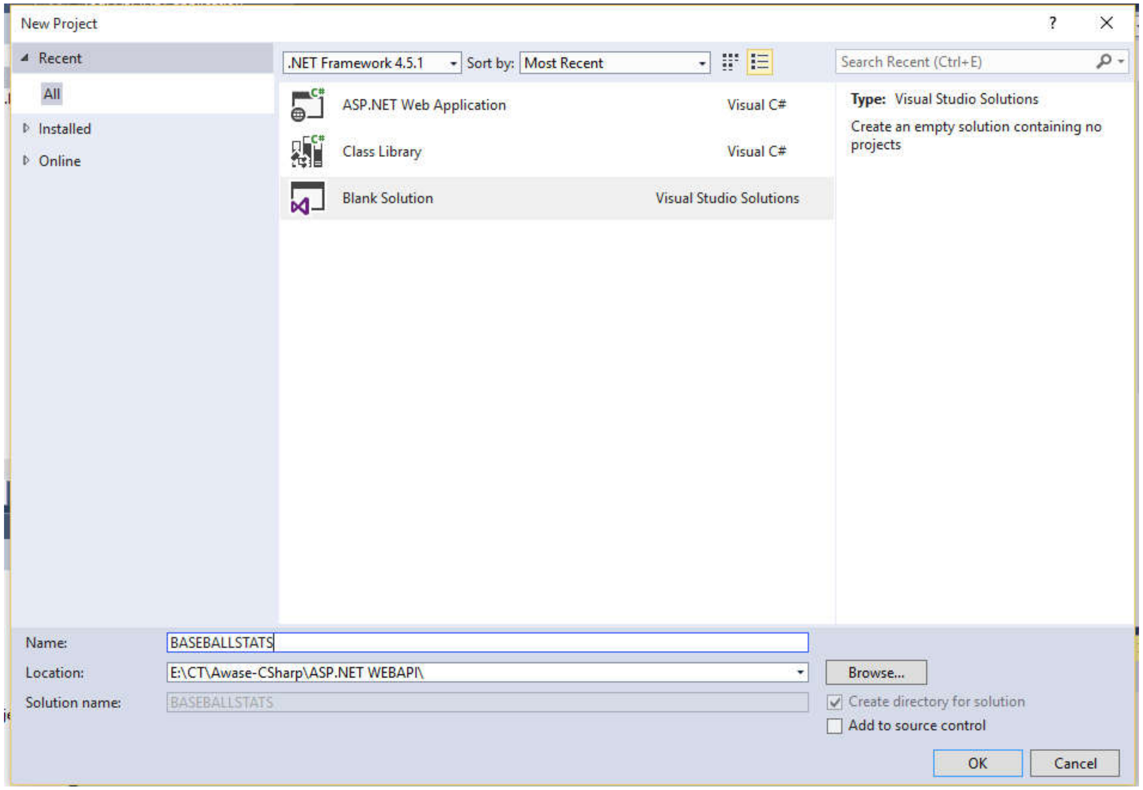
ASP.NET WEB API PROCESSING ARCHITECTURE



SYED AWASE KHIRNI

BASEBALL STATS 3-TIER WEB API APPLICATION DB FIRST APPROACH

C#



.NET Framework 4.5.1 Sort by: Most Recent

ASP.NET Web Application	Visual C#
Class Library	Visual C#
Blank Solution	Visual Studio Solutions

Type: Visual Studio Solutions

Create an empty solution containing no projects

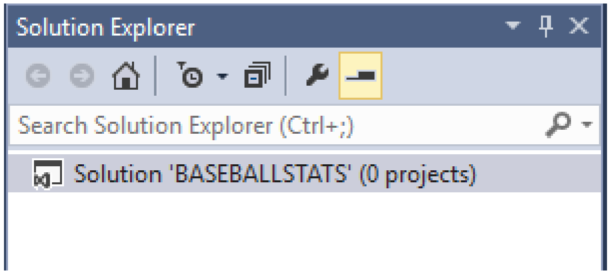
Name: BASEBALLSTATS

Location: E:\CT\Awase-CSharp\ASP.NET WEBAPI\

Solution name: BASEBALLSTATS

☑ Create directory for solution

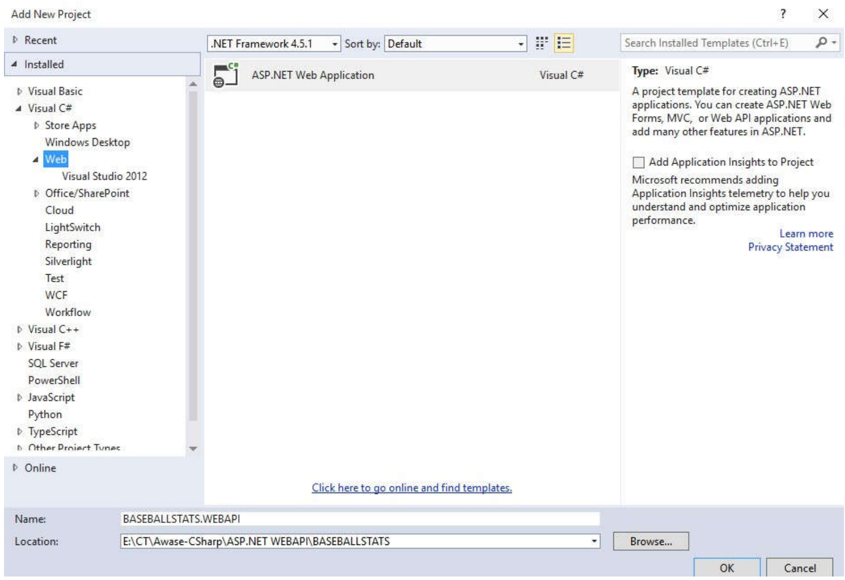
□ Add to source control



Solution 'BASEBALLSTATS' (0 projects)

C#

Create WEB API Project



Add New Project

.NET Framework 4.5.1 Sort by: Default

ASP.NET Web Application Visual C#

Type: Visual C#

A project template for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

□ Add Application Insights to Project

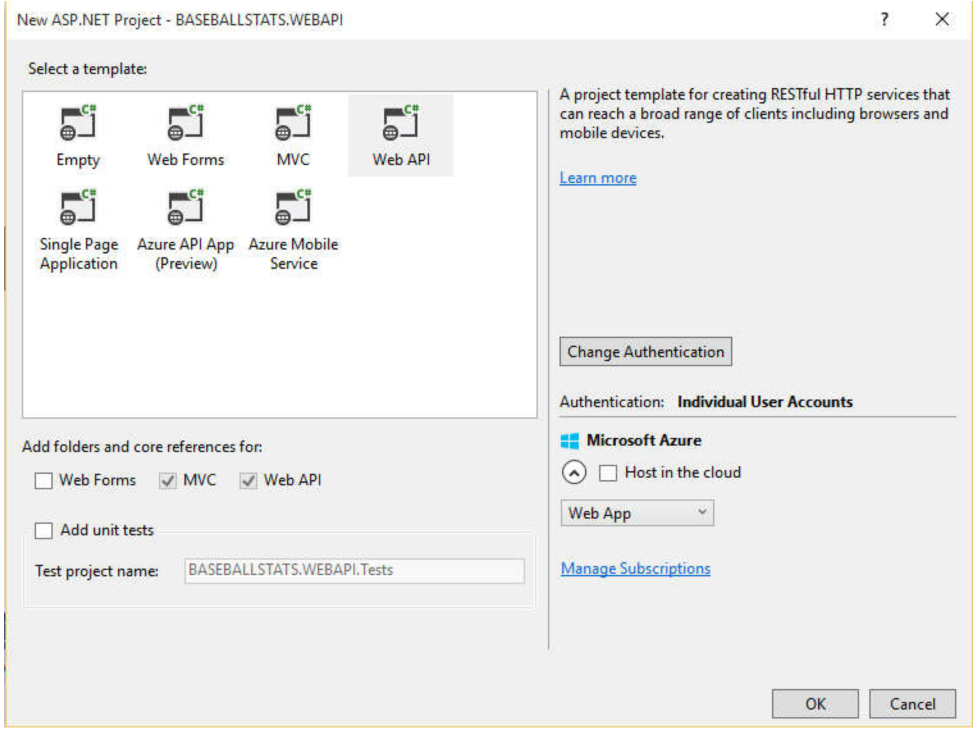
Microsoft recommends adding Application Insights telemetry to help you understand and optimize application performance.

Learn more
Privacy Statement

Click here to go online and find templates.

Name: BASEBALLSTATS.WEBAPI

Location: E:\CT\Awase-CSharp\ASP.NET WEBAPI\BASEBALLSTATS



New ASP.NET Project - BASEBALLSTATS.WEBAPI

Select a template:

A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.

Learn more

Authentication: **Individual User Accounts**

Microsoft Azure

□ Host in the cloud

Manage Subscriptions

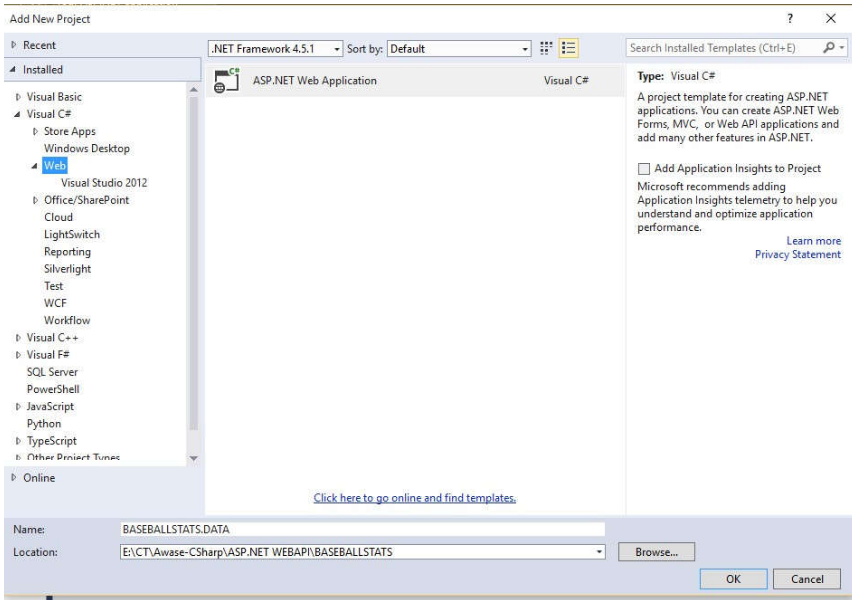
Add folders and core references for:

□ Web Forms ☑ MVC ☑ Web API

□ Add unit tests

Test project name: BASEBALLSTATS.WEBAPI.Tests

Data Layer Project



.NET Framework 4.5.1 Sort by: Default

Type: Visual C#

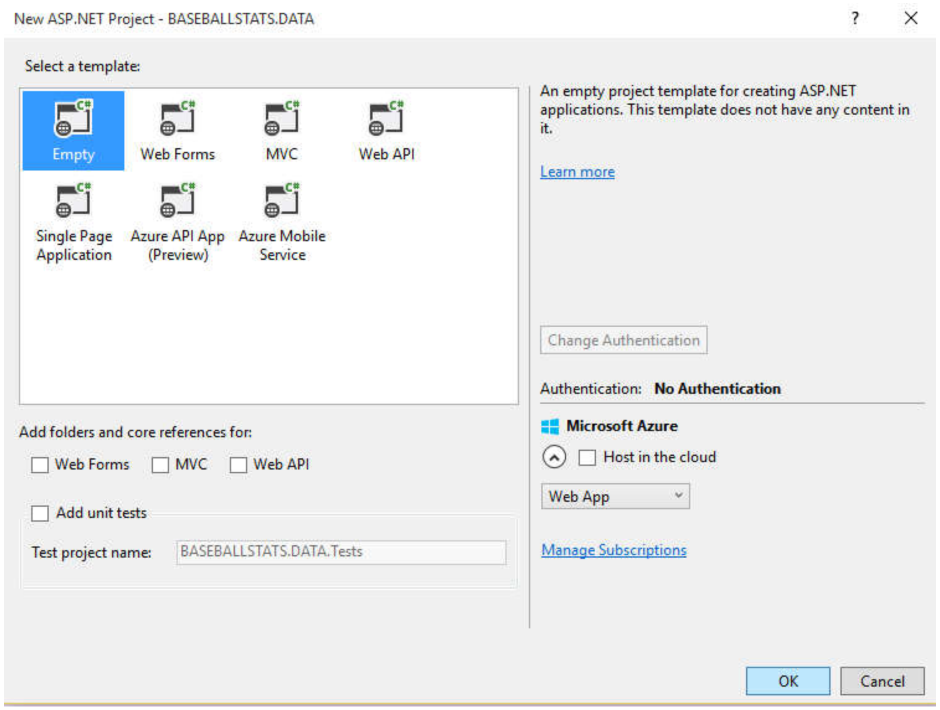
A project template for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

□ Add Application Insights to Project

Microsoft recommends adding Application Insights telemetry to help you understand and optimize application performance.

Name: BASEBALLSTATS.DATA

Location: E:\CT\Awase-CSharp\ASP.NET WEBAPI\BASEBALLSTATS



An empty project template for creating ASP.NET applications. This template does not have any content in it.

Authentication: **No Authentication**

Add folders and core references for:

□ Web Forms □ MVC □ Web API

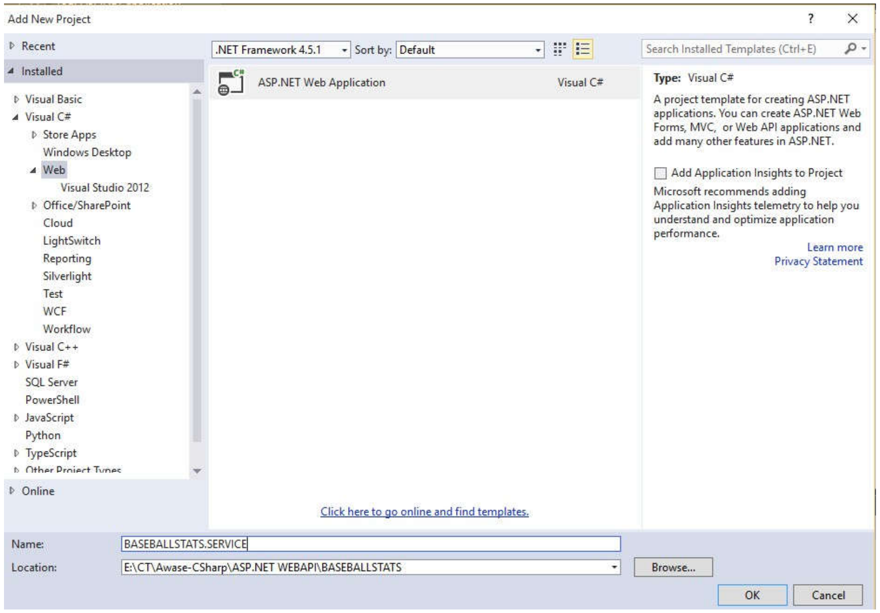
□ Add unit tests

Test project name: BASEBALLSTATS.DATA.Tests

□ Host in the cloud

Web App

Service Layer Project



Type: Visual C#

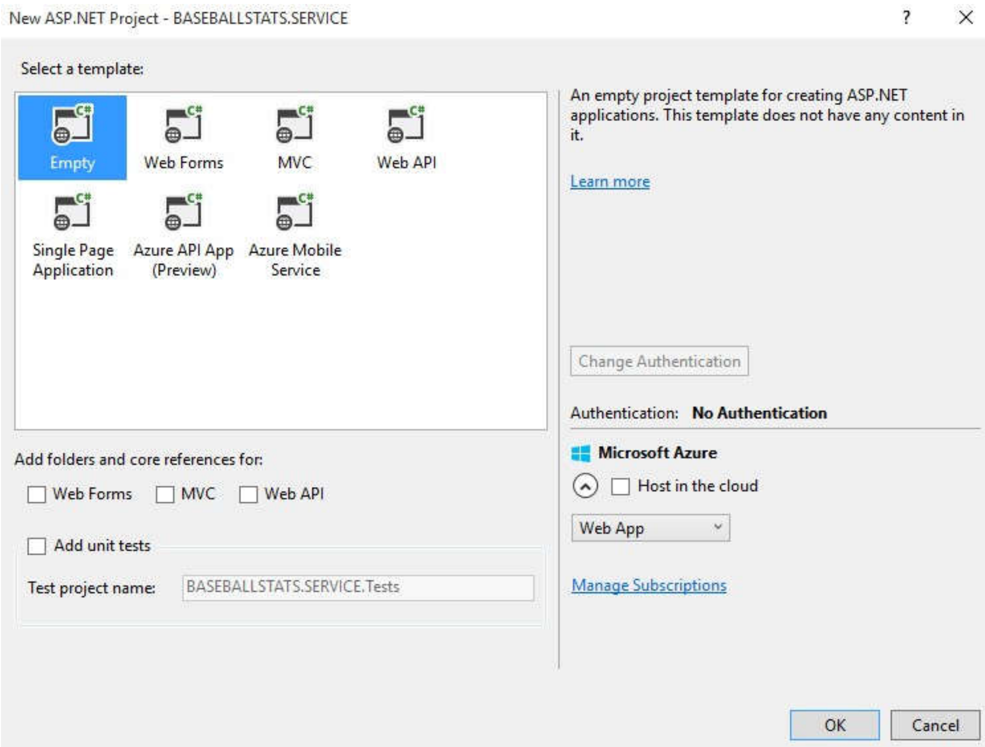
A project template for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

□ Add Application Insights to Project

Microsoft recommends adding Application Insights telemetry to help you understand and optimize application performance.

Name: BASEBALLSTATS.SERVICE

Location: E:\CT\Awase-CSharp\ASP.NET WEBAPI\BASEBALLSTATS



An empty project template for creating ASP.NET applications. This template does not have any content in it.

Authentication: **No Authentication**

Add folders and core references for:

□ Web Forms □ MVC □ Web API

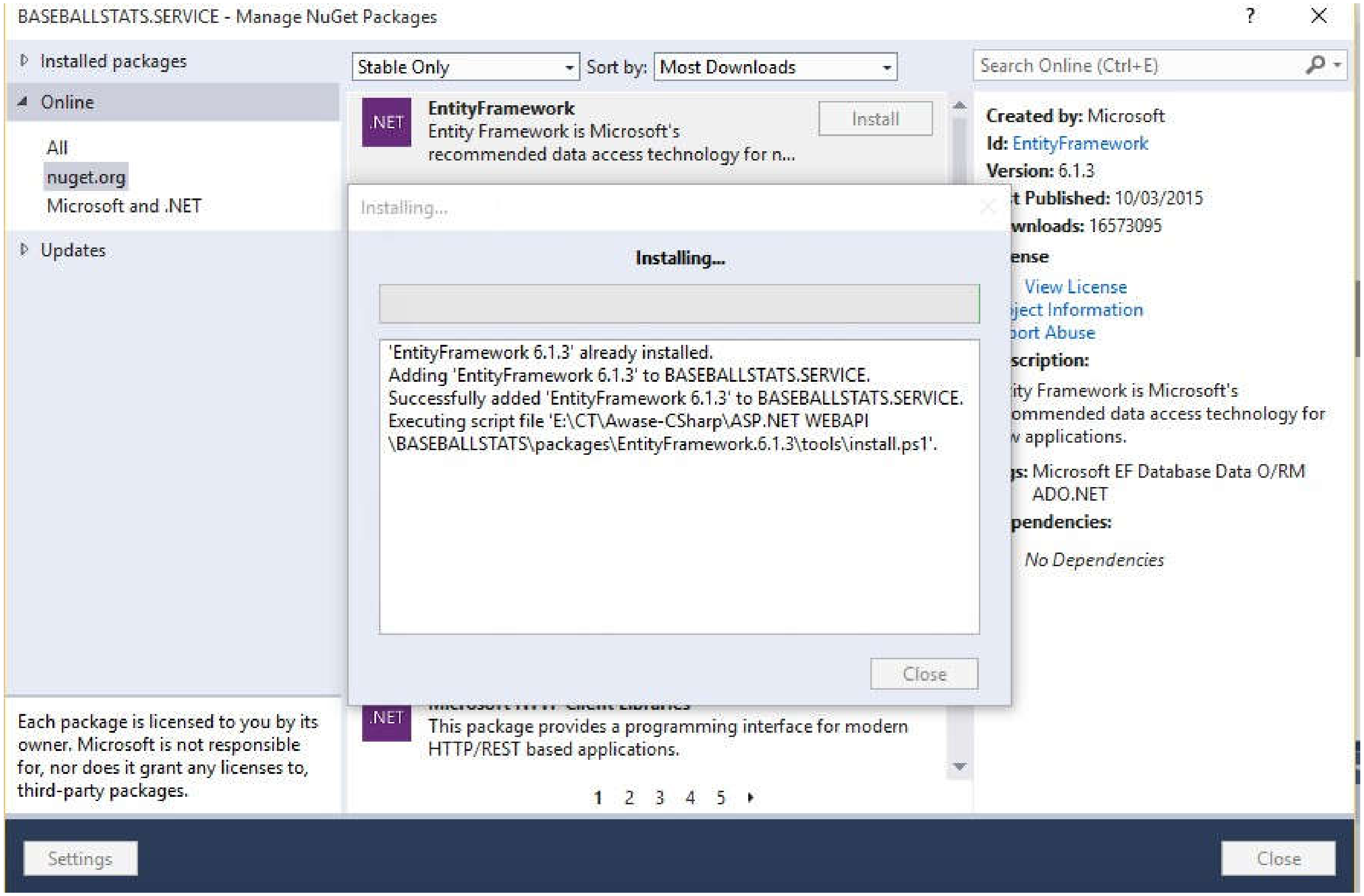
□ Add unit tests

Test project name: BASEBALLSTATS.SERVICE.Tests

□ Host in the cloud

Web App

Install Entity Framework with NuGet to Data and Service Layer Projects



BASEBALLSTATS.SERVICE - Manage NuGet Packages

EntityFramework
Entity Framework is Microsoft's recommended data access technology for n...

Installing...

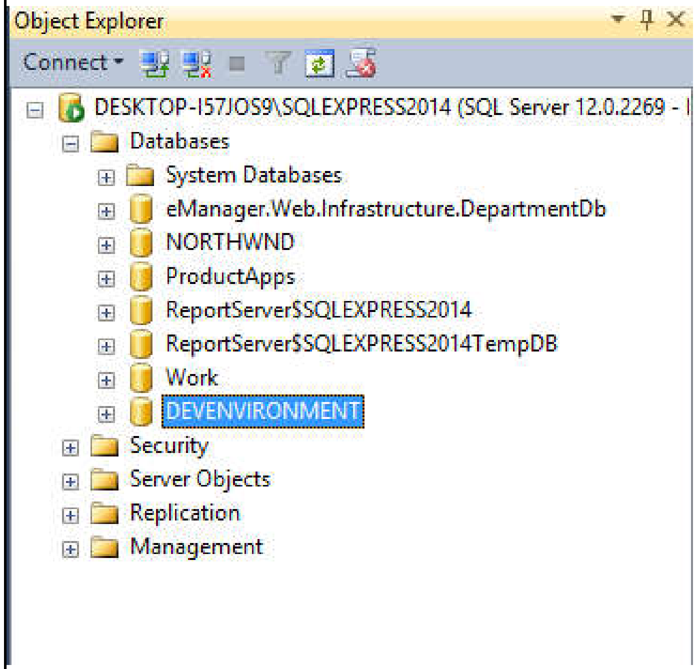
```
'EntityFramework 6.1.3' already installed.
Adding 'EntityFramework 6.1.3' to BASEBALLSTATS.SERVICE.
Successfully added 'EntityFramework 6.1.3' to BASEBALLSTATS.SERVICE.
Executing script file 'E:\CT\Awase-CSharp\ASP.NET WEBAPI
\BASEBALLSTATS\packages\EntityFramework.6.1.3\tools\install.ps1'.
```

Created by: Microsoft
Id: EntityFramework
Version: 6.1.3
t Published: 10/03/2015
wnloads: 16573095
ense
View License
ject Information
port Abuse
scription:
ity Framework is Microsoft's ommended data access technology for v applications.
s: Microsoft EF Database Data O/RM ADO.NET
pendencies:
No Dependencies

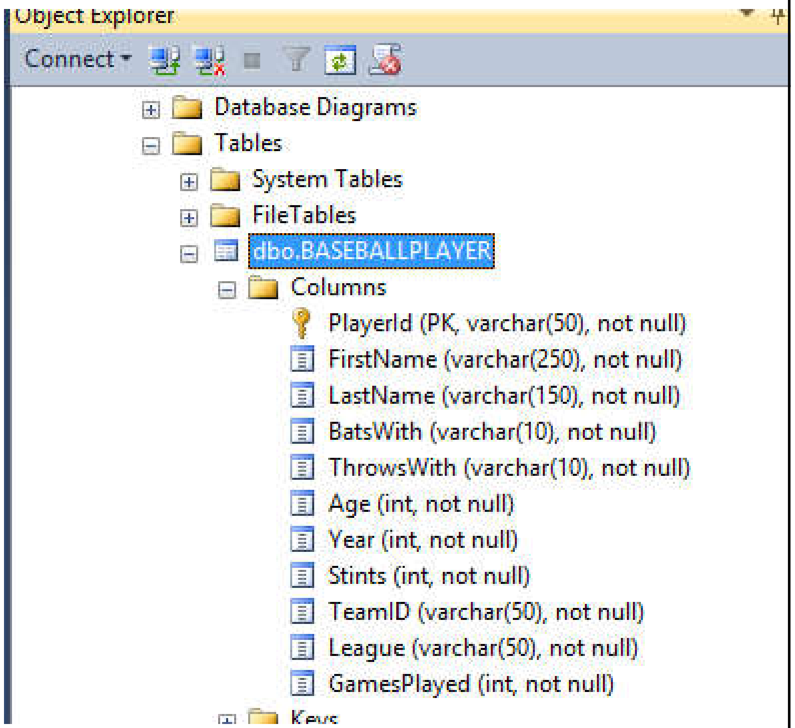
This package provides a programming interface for modern HTTP/REST based applications.

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

Create DB, TABLES



```
Object Explorer
Connect ▾
⊟ DESKTOP-I57JOS9\SQLEXPRESS2014 (SQL Server 12.0.2269 - I
    ⊟ Databases
        ⊞ System Databases
        ⊞ eManager.Web.Infrastructure.DepartmentDb
        ⊞ NORTHWND
        ⊞ ProductApps
        ⊞ ReportServer$SQLEXPRESS2014
        ⊞ ReportServer$SQLEXPRESS2014TempDB
        ⊞ Work
        ⊞ DEVENVIRONMENT
    ⊞ Security
    ⊞ Server Objects
    ⊞ Replication
    ⊞ Management
```

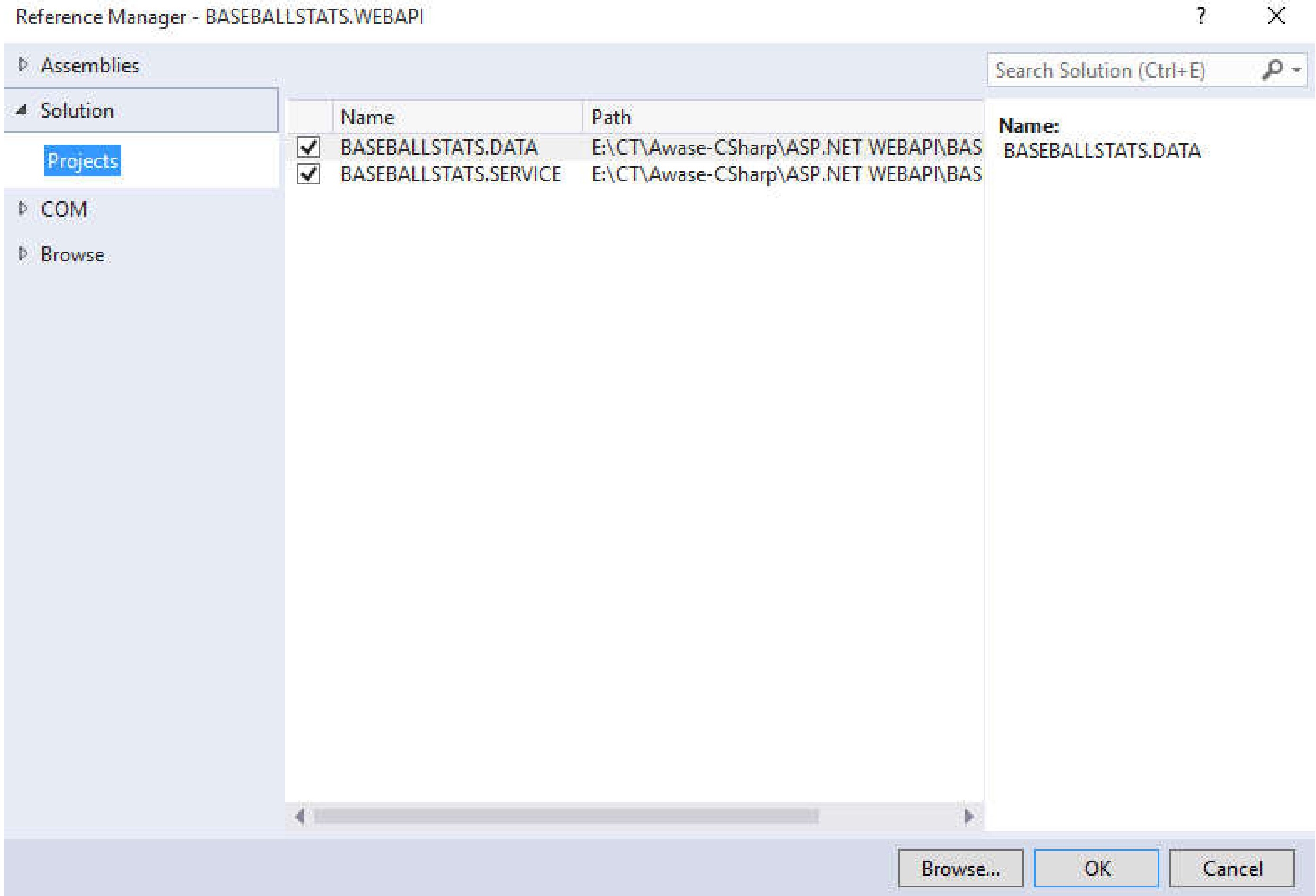


```
Object Explorer
Connect ▾
    ⊞ Database Diagrams
    ⊟ Tables
        ⊞ System Tables
        ⊞ FileTables
        ⊟ dbo.BASEBALLPLAYER
            ⊟ Columns
                PlayerId (PK, varchar(50), not null)
                FirstName (varchar(250), not null)
                LastName (varchar(150), not null)
                BatsWith (varchar(10), not null)
                ThrowsWith (varchar(10), not null)
                Age (int, not null)
                Year (int, not null)
                Stints (int, not null)
                TeamID (varchar(50), not null)
                League (varchar(50), not null)
                GamesPlayed (int, not null)
            ⊞ Keys
```

Populate data through Insert Statements in SQL Query

C#

ADD REFERENCE FOR WEB API LAYER

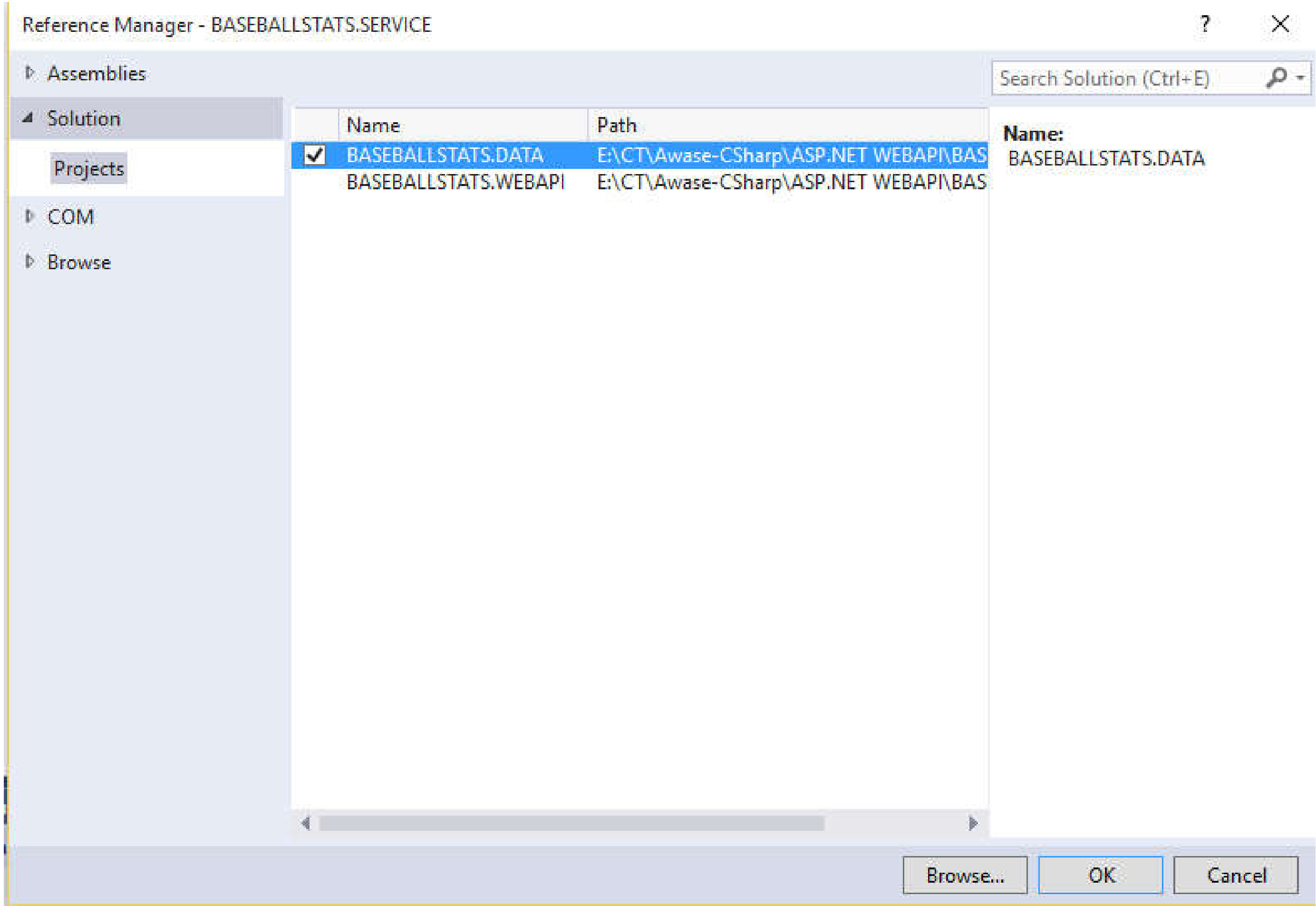


Reference Manager - BASEBALLSTATS.WEBAPI

	Name	Path
☑	BASEBALLSTATS.DATA	E:\CT\Awase-CSharp\ASP.NET WEBAPI\BAS
☑	BASEBALLSTATS.SERVICE	E:\CT\Awase-CSharp\ASP.NET WEBAPI\BAS

Name: BASEBALLSTATS.DATA

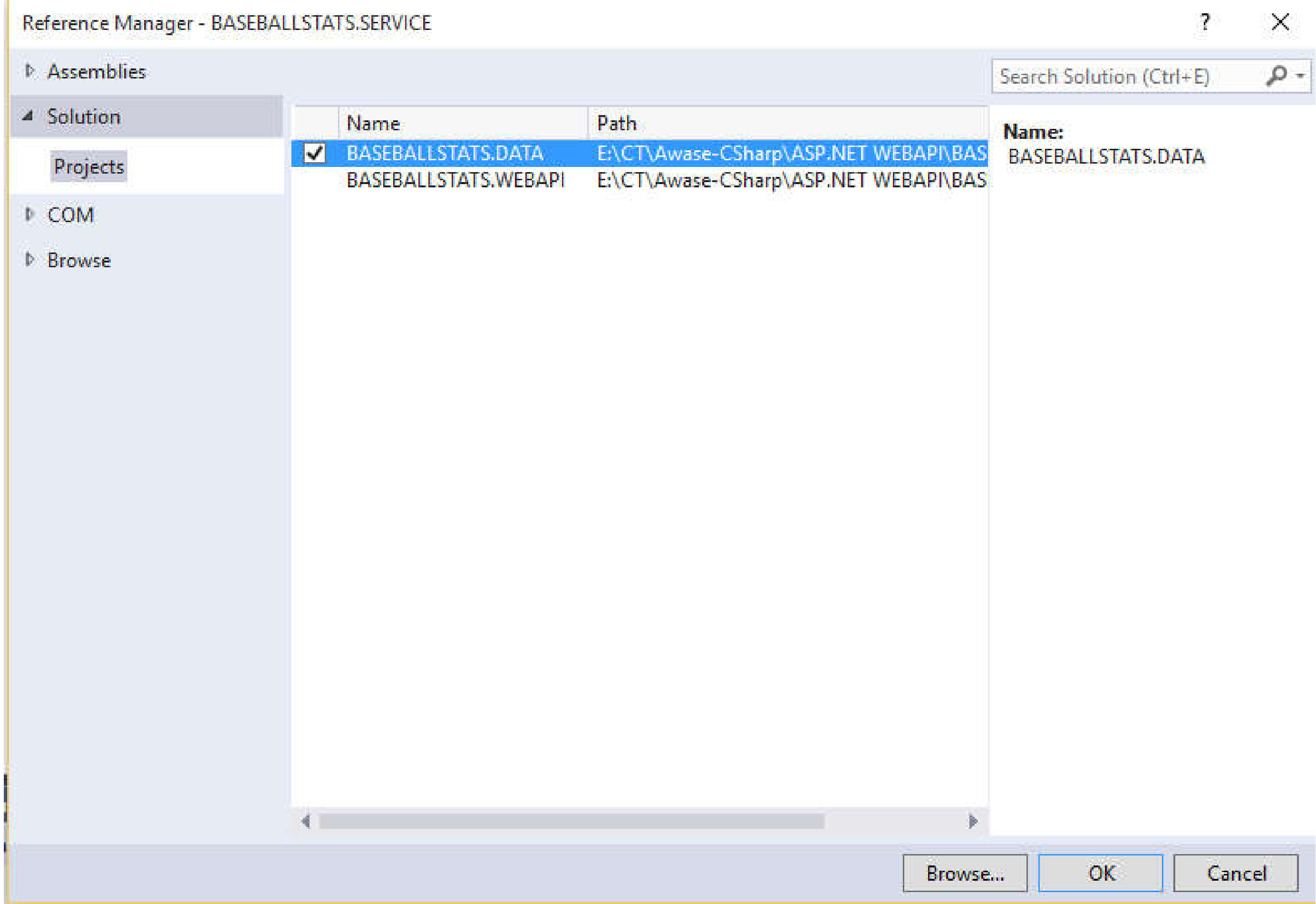
ADD REFERENCE FOR SERVICE LAYER



	Name	Path
☑	BASEBALLSTATS.DATA	E:\CT\Awase-CSharp\ASP.NET WEBAPI\BAS
	BASEBALLSTATS.WEBAPI	E:\CT\Awase-CSharp\ASP.NET WEBAPI\BAS

Name:
BASEBALLSTATS.DATA

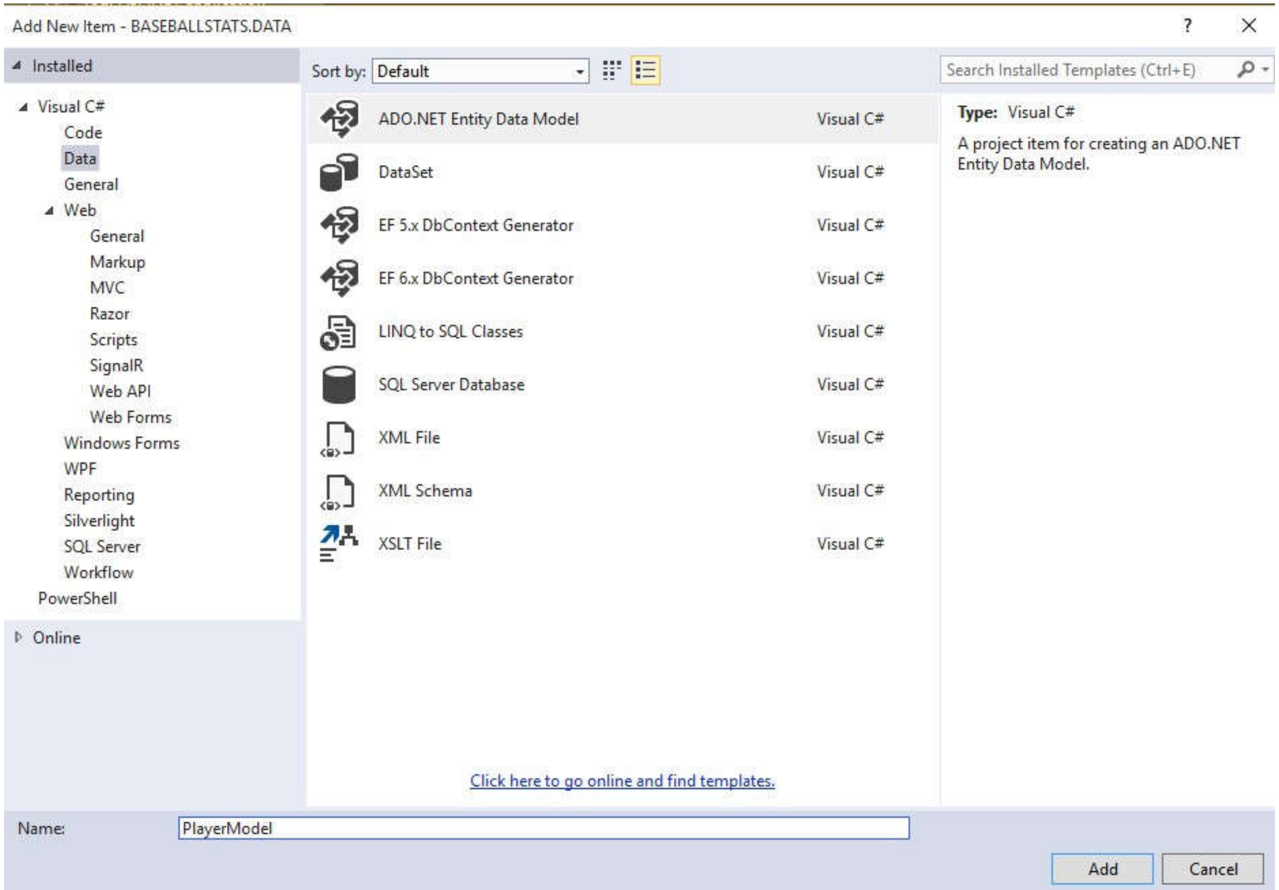
DATA LAYER EF



	Name	Path
☑	BASEBALLSTATS.DATA	E:\CT\Awase-CSharp\ASP.NET WEBAPI\BAS
	BASEBALLSTATS.WEBAPI	E:\CT\Awase-CSharp\ASP.NET WEBAPI\BAS

Name:
BASEBALLSTATS.DATA

EF DB FIRST



Add New Item - BASEBALLSTATS.DATA

Sort by: Default

ADO.NET Entity Data Model	Visual C#
DataSet	Visual C#
EF 5.x DbContext Generator	Visual C#
EF 6.x DbContext Generator	Visual C#
LINQ to SQL Classes	Visual C#
SQL Server Database	Visual C#
XML File	Visual C#
XML Schema	Visual C#
XSLT File	Visual C#

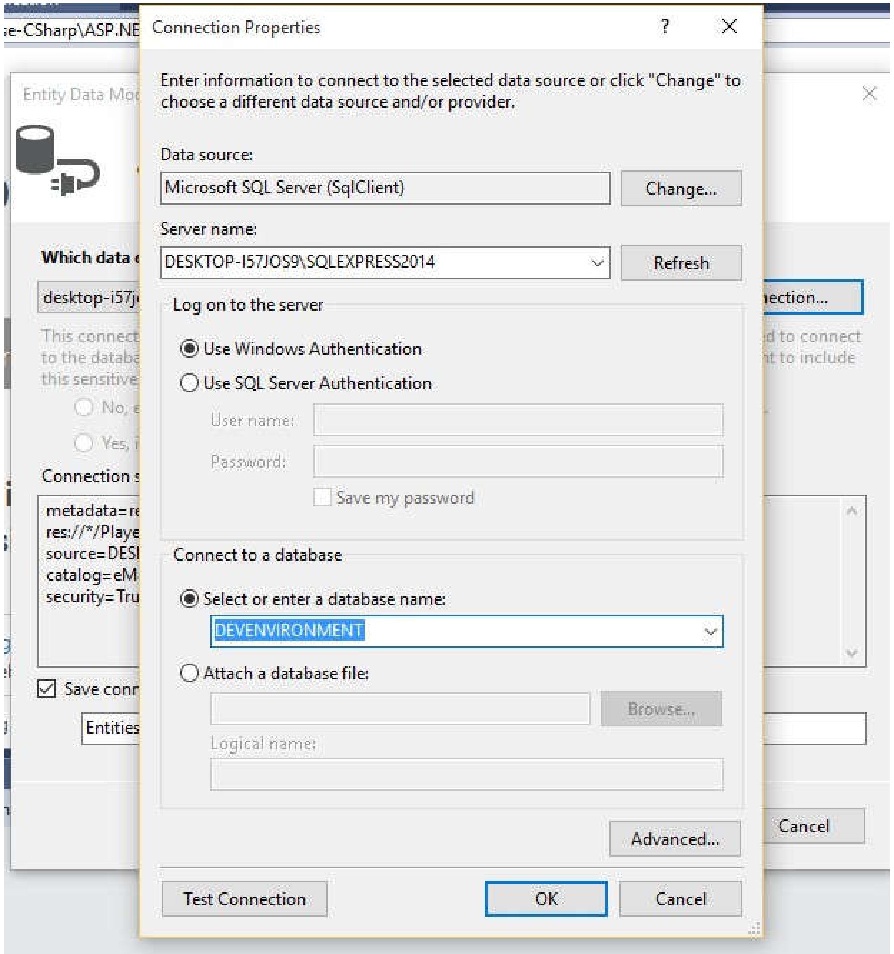
Type: Visual C#

A project item for creating an ADO.NET Entity Data Model.

Name: PlayerModel

C#

DB Connection Settings



Connection Properties

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: DESKTOP-I57JOS9\SQLEXPRESS2014

Log on to the server

● Use Windows Authentication

○ Use SQL Server Authentication

User name:

Password:

□ Save my password

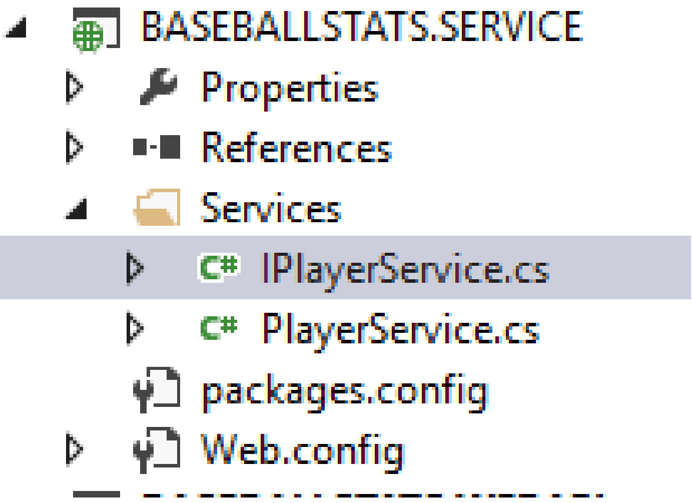
Connect to a database

● Select or enter a database name: DEVENVIRONMENT

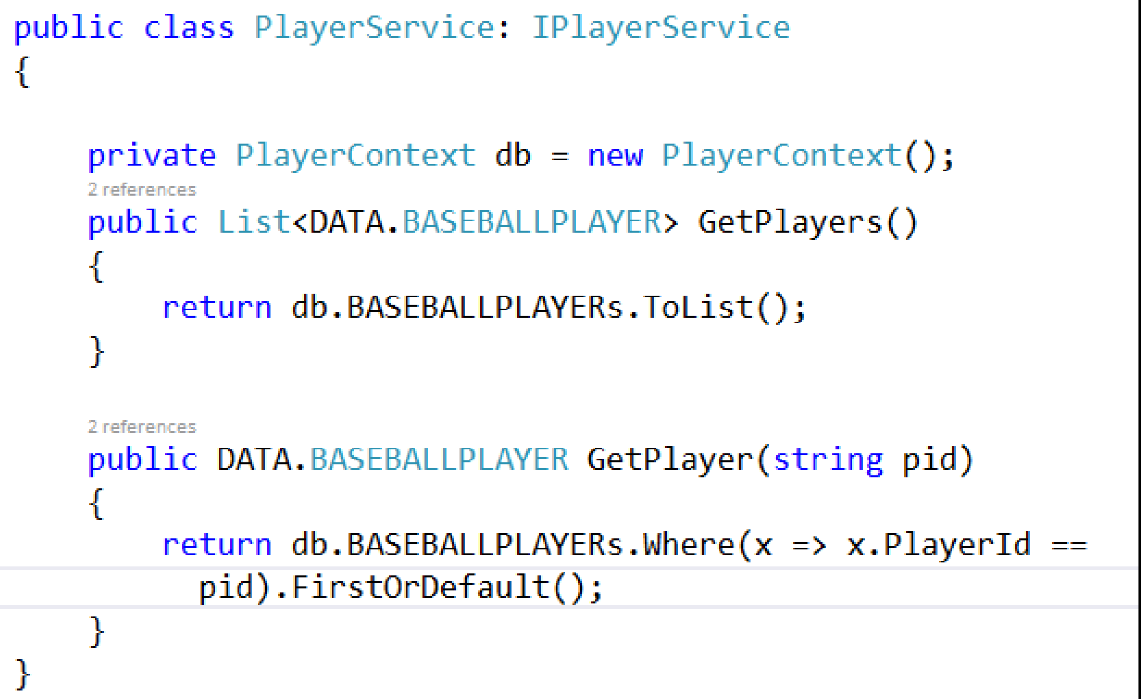
○ Attach a database file:

Logical name:

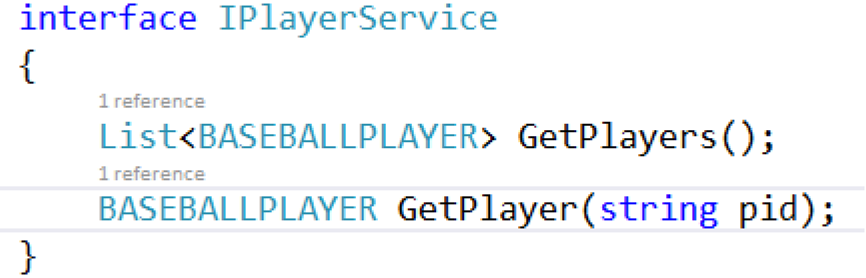
Service Layer



```
◢ BASEBALLSTATS.SERVICE
  ▷ Properties
  ▷ References
  ◢ Services
    ▷ IPlayerService.cs
    ▷ PlayerService.cs
    packages.config
  ▷ Web.config
```

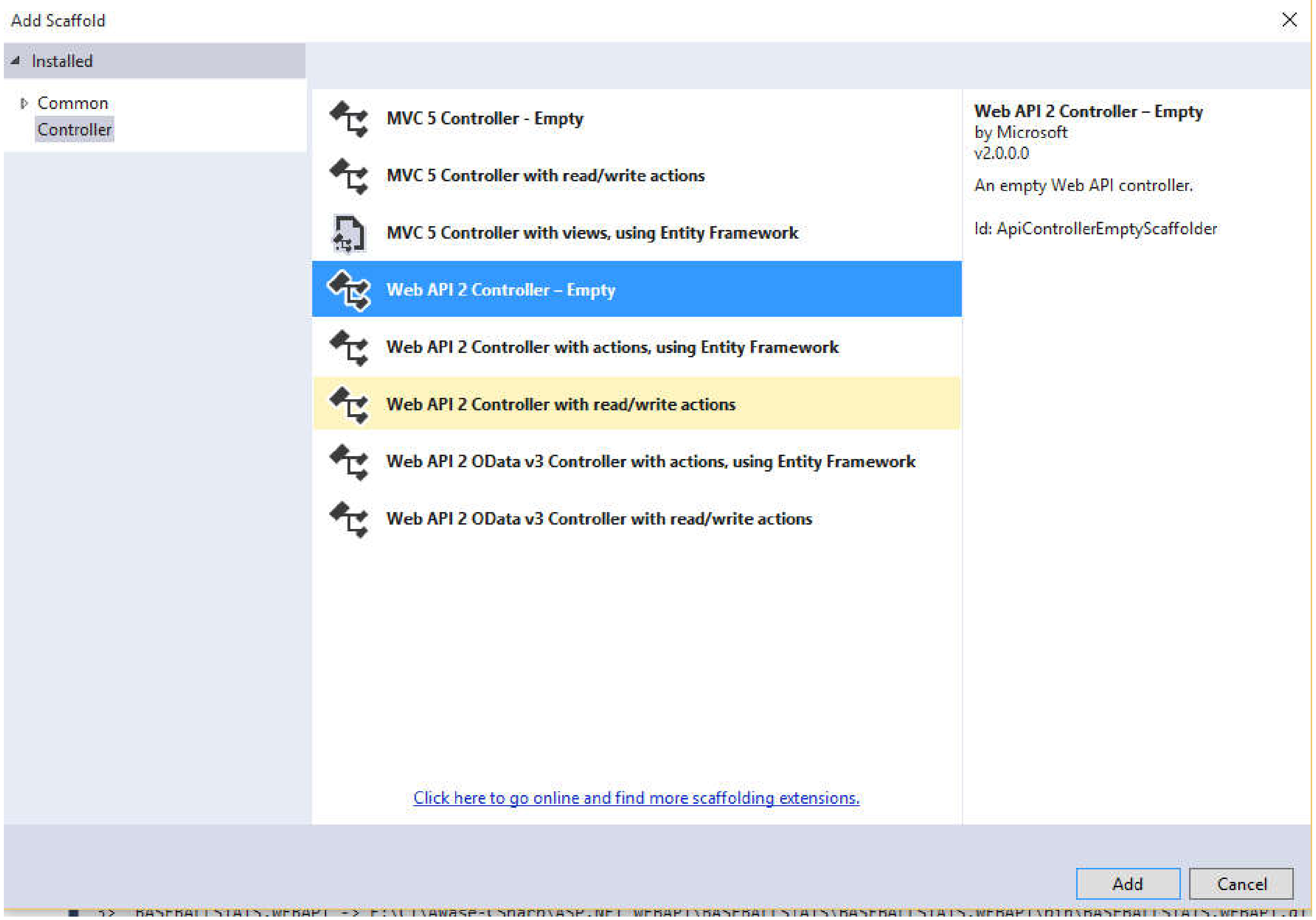


```
public class PlayerService: IPlayerService
{

    private PlayerContext db = new PlayerContext();
    2 references
    public List<DATA.BASEBALLPLAYER> GetPlayers()
    {
        return db.BASEBALLPLAYERs.ToList();
    }

    2 references
    public DATA.BASEBALLPLAYER GetPlayer(string pid)
    {
        return db.BASEBALLPLAYERs.Where(x => x.PlayerId ==
          pid).FirstOrDefault();
    }
}
```



```
1 reference
interface IPlayerService
{
    1 reference
    List<BASEBALLPLAYER> GetPlayers();
    1 reference
    BASEBALLPLAYER GetPlayer(string pid);
}
```

Add WebAPI Controller



Add Scaffold

MVC 5 Controller - Empty

MVC 5 Controller with read/write actions

MVC 5 Controller with views, using Entity Framework

Web API 2 Controller – Empty

Web API 2 Controller with actions, using Entity Framework

Web API 2 Controller with read/write actions

Web API 2 OData v3 Controller with actions, using Entity Framework

Web API 2 OData v3 Controller with read/write actions

Click here to go online and find more scaffolding extensions.

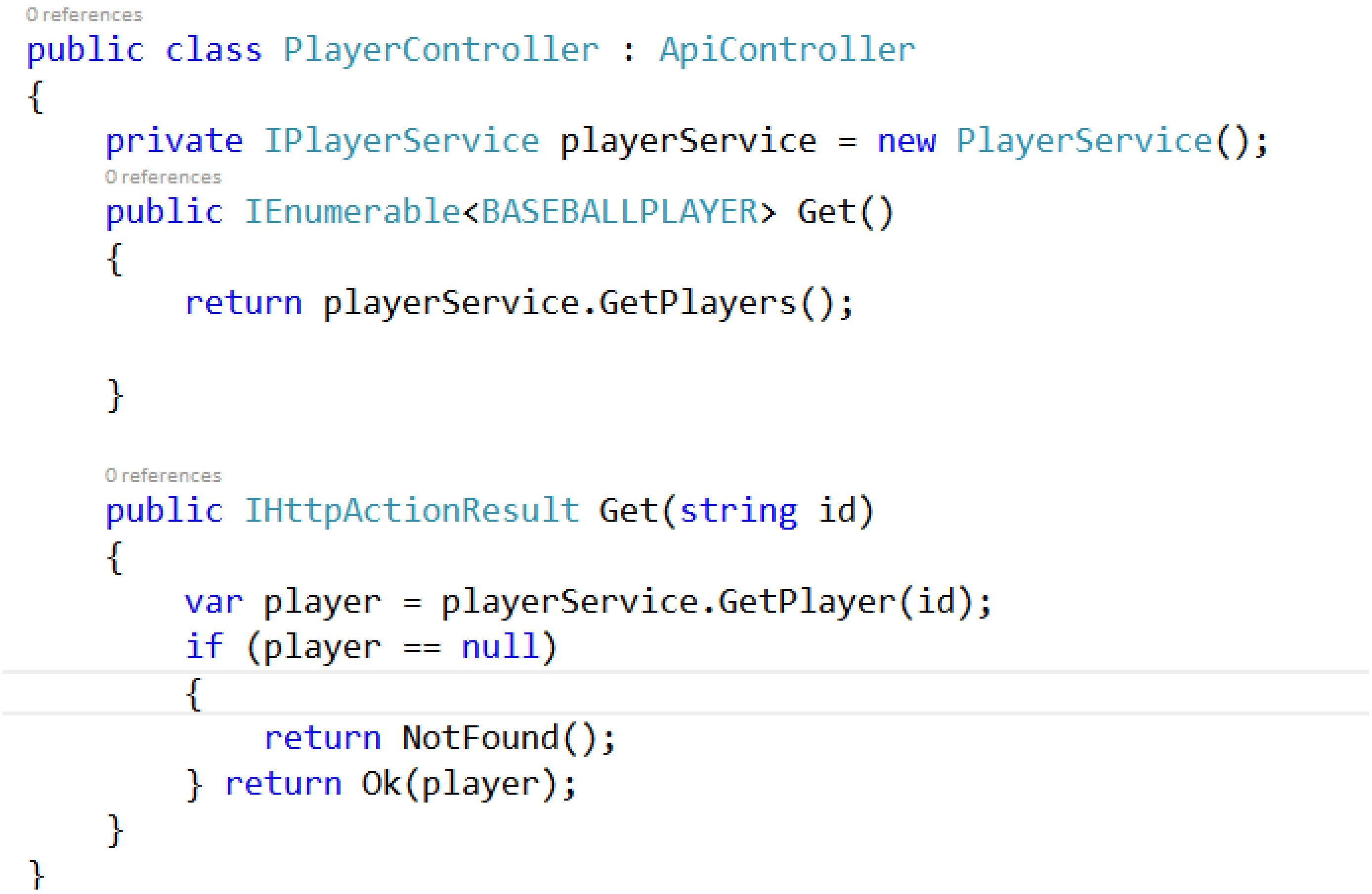
Web API 2 Controller – Empty
by Microsoft
v2.0.0.0

An empty Web API controller.

Id: ApiControllerEmptyScaffolder

```
1> BASEBALLSTATS.WEBAPI -> F:\CT\Awase-CSharp\ASP.NET WEBAPI\BASEBALLSTATS\BASEBALLSTATS.WEBAPI\bin\BASEBALLSTATS.WEBAPI.dl
```

ApiController



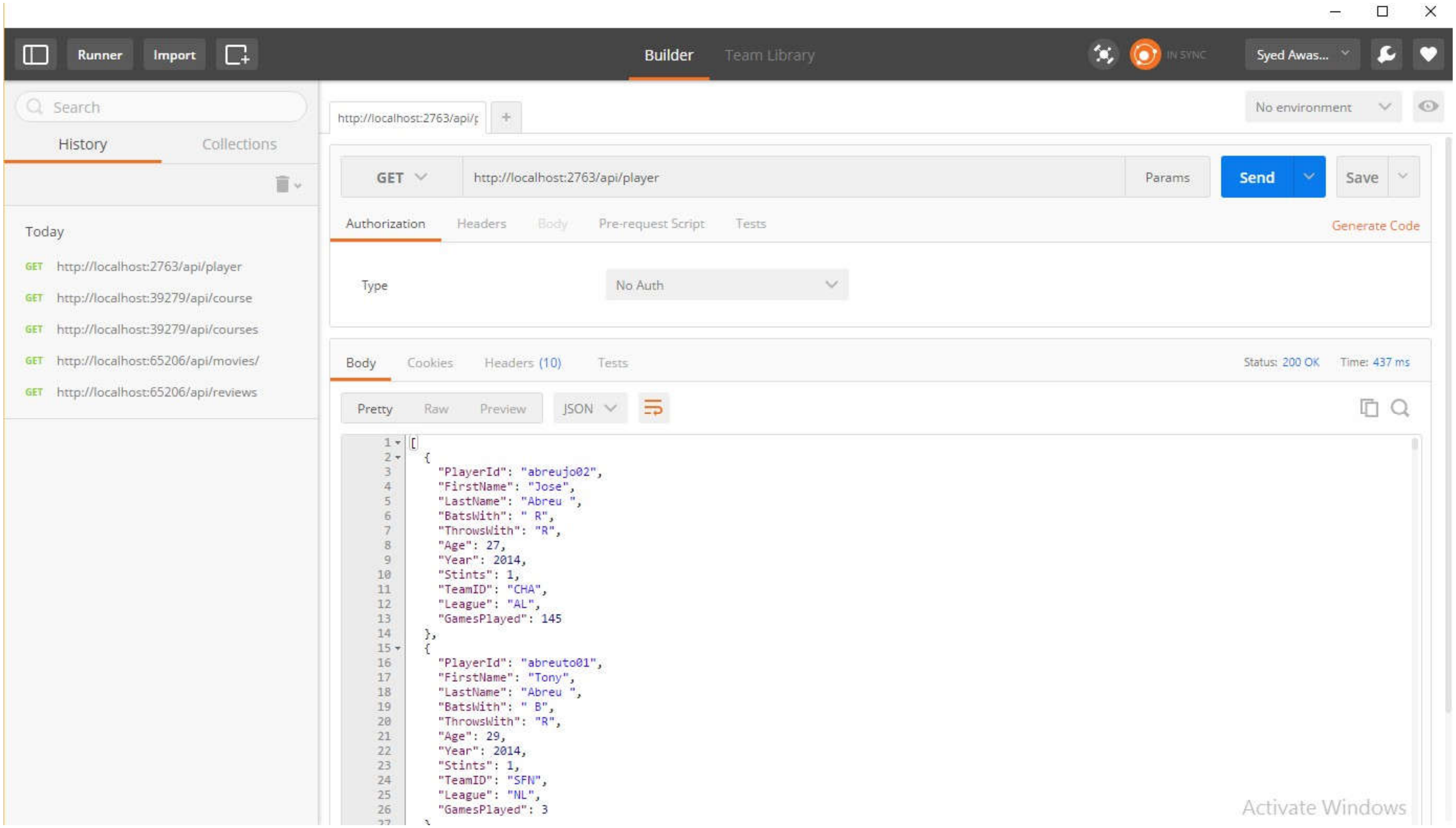
```
0 references
public class PlayerController : ApiController
{
    private IPlayerService playerService = new PlayerService();
    0 references
    public IEnumerable<BASEBALLPLAYER> Get()
    {
        return playerService.GetPlayers();

    }

    0 references
    public IHttpActionResult Get(string id)
    {
        var player = playerService.GetPlayer(id);
        if (player == null)
        {
            return NotFound();
        } return Ok(player);
    }
}
```

Add Connection String

WebAPI -> Web.config

```
<connectionStrings>
   <add name="PlayerContext"
connectionString="metadata=res://*/PlayerModel.csdl|res://*/PlayerModel.ss
dl|res://*/PlayerModel.msl;provider=System.Data.SqlClient;provider
connection string=&quot;data source=DESKTOP-
I57JOS9\SQLEXPRESS2014;initial catalog=DEVENVIRONMENT;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework&quot;"
providerName="System.Data.EntityClient" />
  </connectionStrings>
```

C#



History:

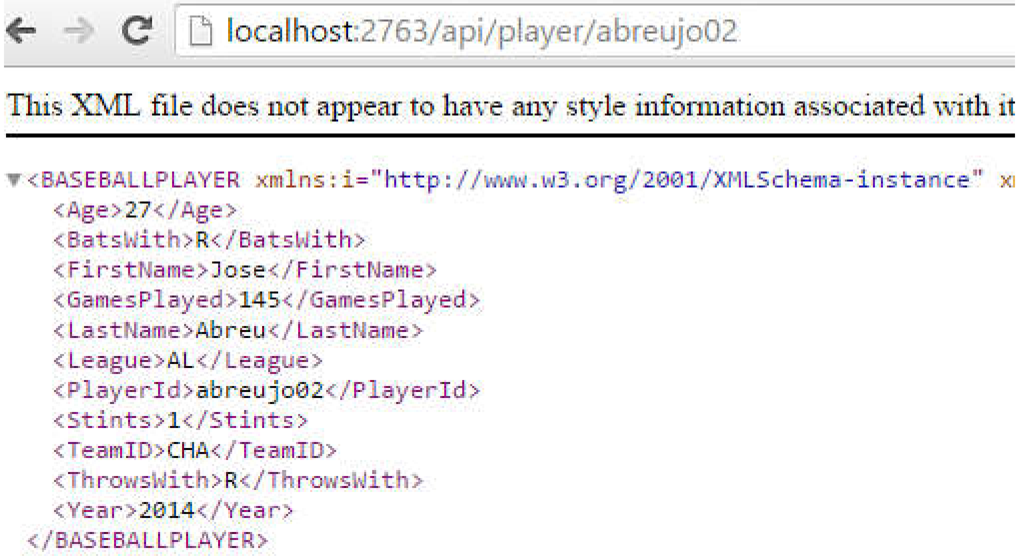
- GET http://localhost:2763/api/player
- GET http://localhost:39279/api/course
- GET http://localhost:39279/api/courses
- GET http://localhost:65206/api/movies/
- GET http://localhost:65206/api/reviews

GET http://localhost:2763/api/player

Type: No Auth

Status: 200 OK   Time: 437 ms

```
[
  {
    "PlayerId": "abreujo02",
    "FirstName": "Jose",
    "LastName": "Abreu ",
    "BatsWith": " R",
    "ThrowsWith": "R",
    "Age": 27,
    "Year": 2014,
    "Stints": 1,
    "TeamID": "CHA",
    "League": "AL",
    "GamesPlayed": 145
  },
  {
    "PlayerId": "abreuto01",
    "FirstName": "Tony",
    "LastName": "Abreu ",
    "BatsWith": " B",
    "ThrowsWith": "R",
    "Age": 29,
    "Year": 2014,
    "Stints": 1,
    "TeamID": "SFN",
    "League": "NL",
    "GamesPlayed": 3
  }
```



localhost:2763/api/player/abreujo02

This XML file does not appear to have any style information associated with it

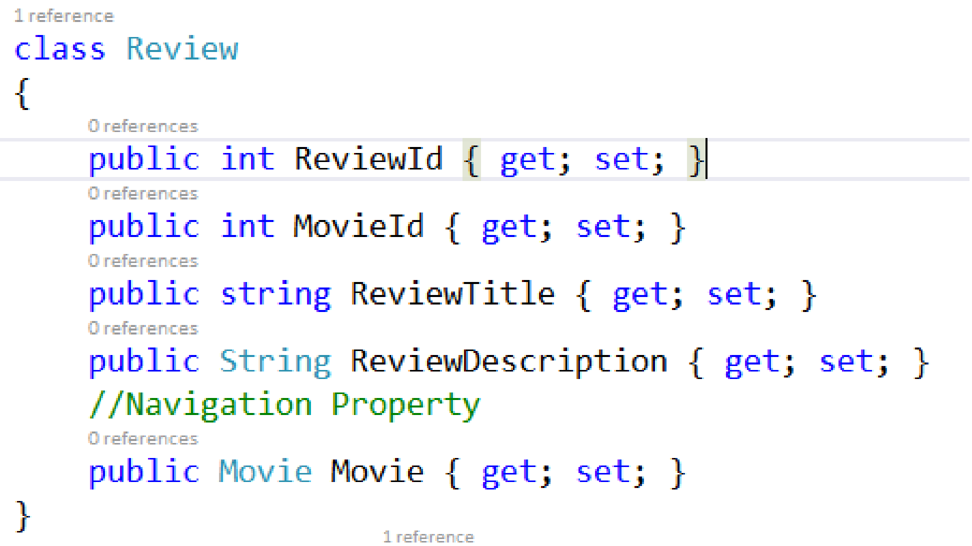
```
▼<BASEBALLPLAYER xmlns:i="http://www.w3.org/2001/XMLSchema-instance" x
   <Age>27</Age>
   <BatsWith>R</BatsWith>
   <FirstName>Jose</FirstName>
   <GamesPlayed>145</GamesPlayed>
   <LastName>Abreu</LastName>
   <League>AL</League>
   <PlayerId>abreujo02</PlayerId>
   <Stints>1</Stints>
   <TeamID>CHA</TeamID>
   <ThrowsWith>R</ThrowsWith>
   <Year>2014</Year>
 </BASEBALLPLAYER>
```

SYED AWASE KHIRNI

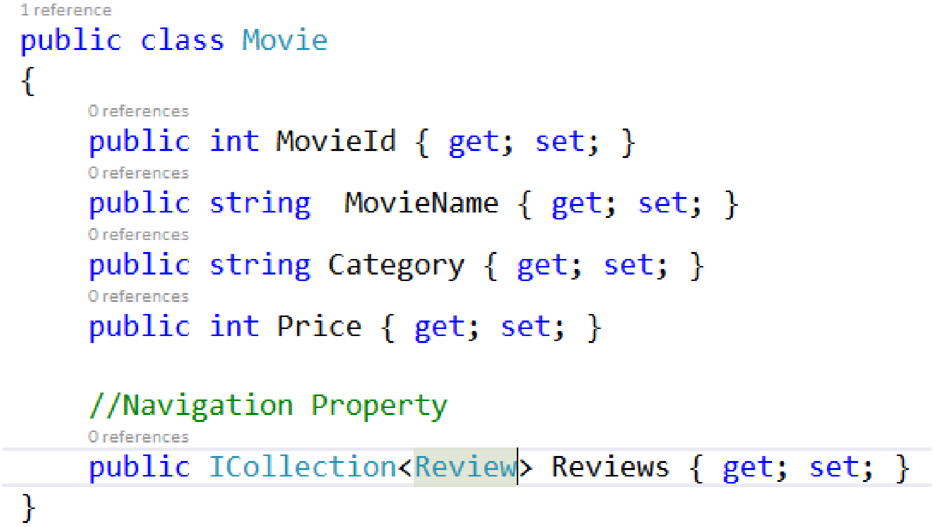
MOVIE REVIEW WEB API APPLICATION WITH CODE FIRST APPROACH

C#

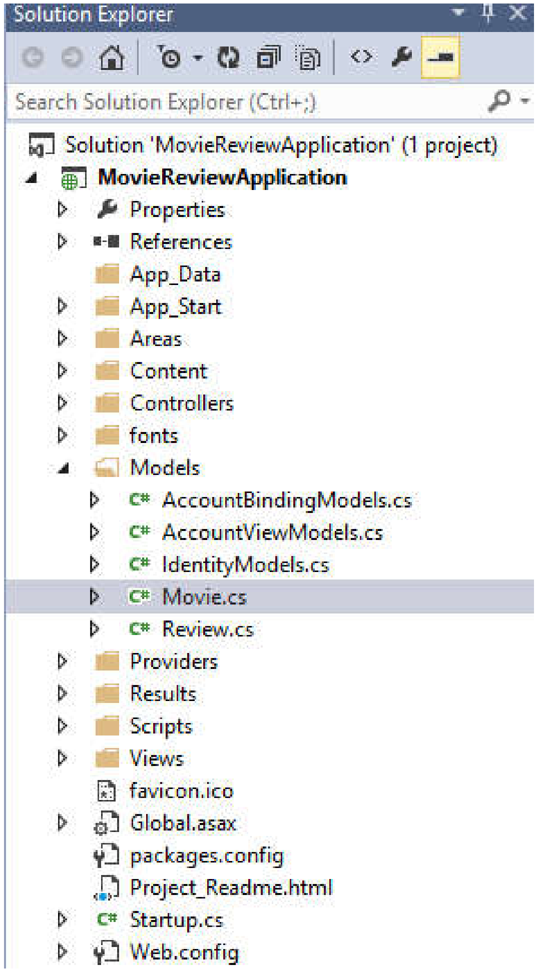
Movie and Review Classes



```
1 reference
class Review
{
    0 references
    public int ReviewId { get; set; }
    0 references
    public int MovieId { get; set; }
    0 references
    public string ReviewTitle { get; set; }
    0 references
    public String ReviewDescription { get; set; }
    //Navigation Property
    0 references
    public Movie Movie { get; set; }
}
```



```
1 reference
public class Movie
{
    0 references
    public int MovieId { get; set; }
    0 references
    public string  MovieName { get; set; }
    0 references
    public string Category { get; set; }
    0 references
    public int Price { get; set; }

    //Navigation Property
    0 references
    public ICollection<Review> Reviews { get; set; }
}
```

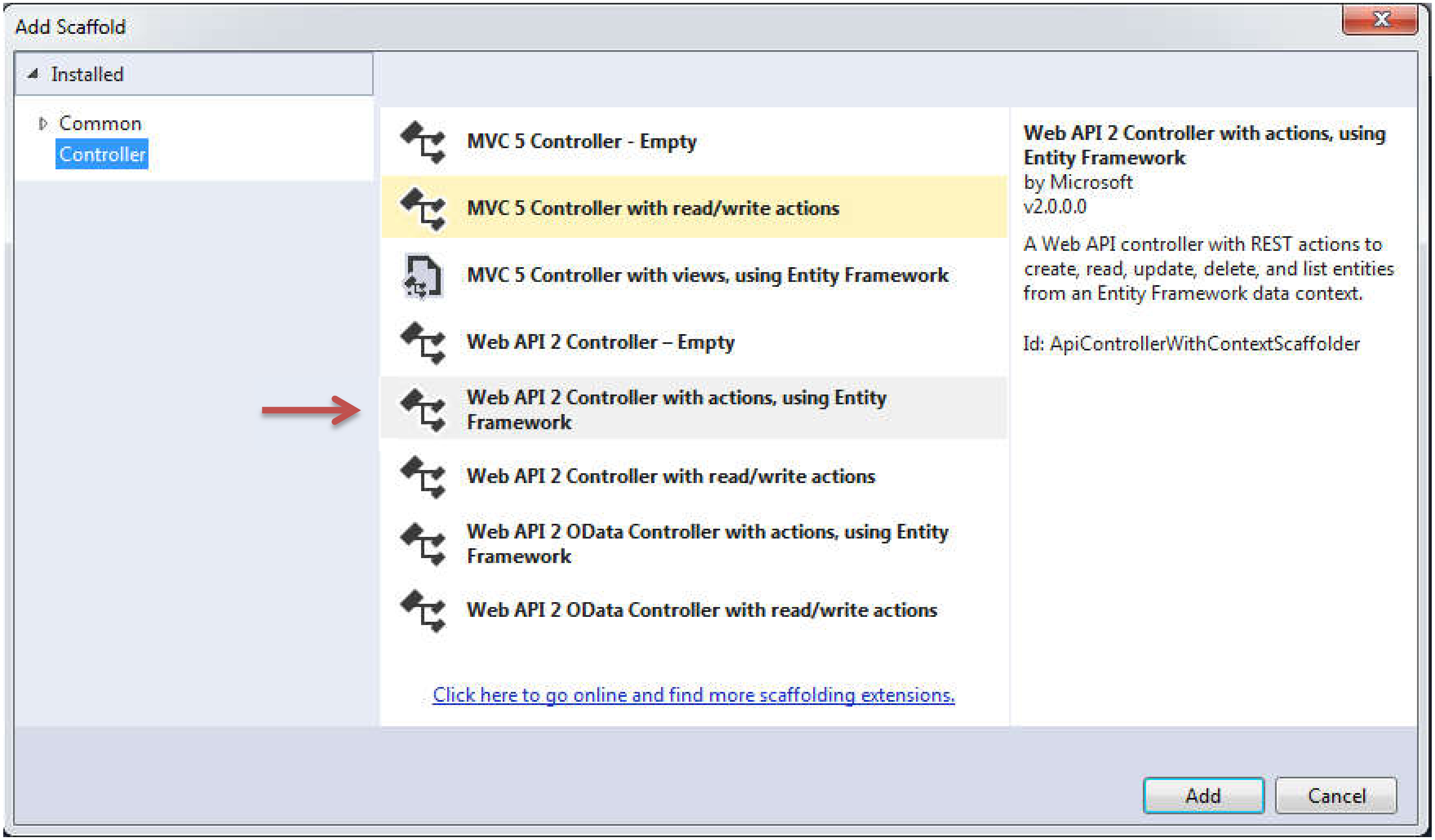


```
Solution 'MovieReviewApplication' (1 project)
  MovieReviewApplication
    Properties
    References
    App_Data
    App_Start
    Areas
    Content
    Controllers
    fonts
    Models
      AccountBindingModels.cs
      AccountViewModels.cs
      IdentityModels.cs
      Movie.cs
      Review.cs
    Providers
    Results
    Scripts
    Views
    favicon.ico
    Global.asax
    packages.config
    Project_Readme.html
    Startup.cs
    Web.config
```

Connection String

Web.config

```
<connectionStrings>
<add name="DefaultConnection" connectionString="Data Source=DESKTOP-
I57JOS9\SQLEXPRESS2014;Initial Catalog=Work;Integrated Security = SSPI"
providerName="System.Data.SqlClient" />
</connectionStrings>
```



Add Scaffold

- Installed
  - Common
  - Controller

- MVC 5 Controller - Empty
- MVC 5 Controller with read/write actions
- MVC 5 Controller with views, using Entity Framework
- Web API 2 Controller – Empty
- Web API 2 Controller with actions, using Entity Framework
- Web API 2 Controller with read/write actions
- Web API 2 OData Controller with actions, using Entity Framework
- Web API 2 OData Controller with read/write actions

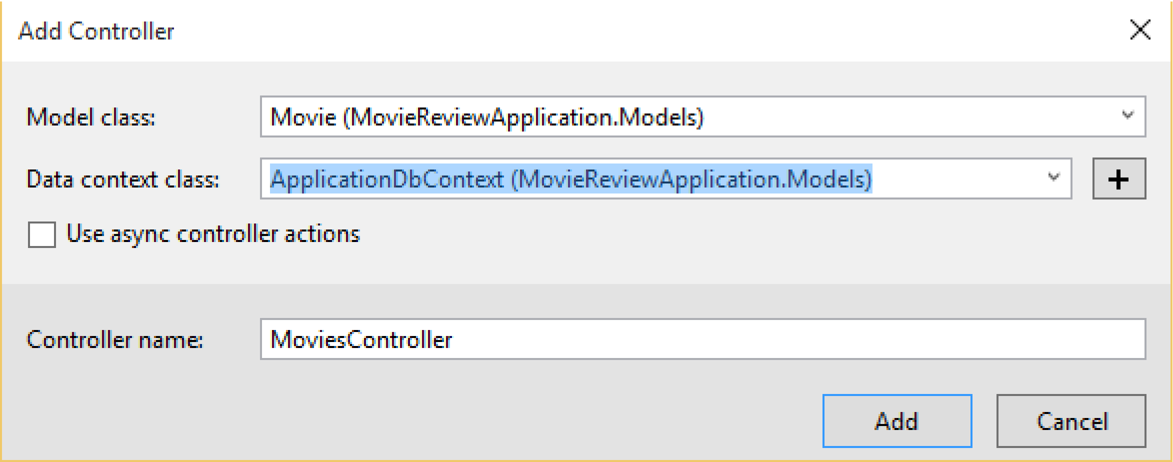
Click here to go online and find more scaffolding extensions.

Web API 2 Controller with actions, using Entity Framework
by Microsoft
v2.0.0.0

A Web API controller with REST actions to create, read, update, delete, and list entities from an Entity Framework data context.

Id: ApiControllerWithContextScaffolder

C#



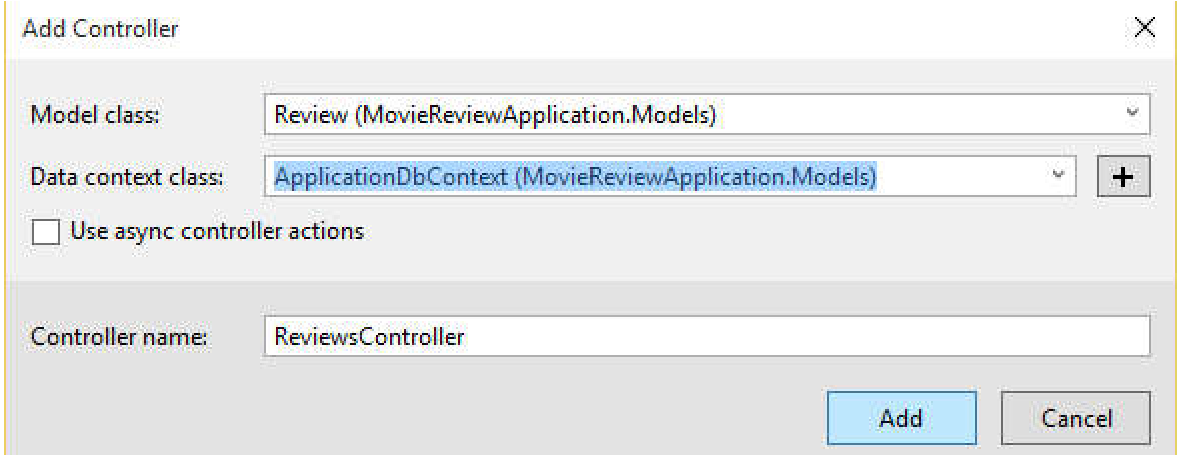
Add Controller

Model class: Movie (MovieReviewApplication.Models)

Data context class: ApplicationDbContext (MovieReviewApplication.Models)

□ Use async controller actions

Controller name: MoviesController



Add Controller

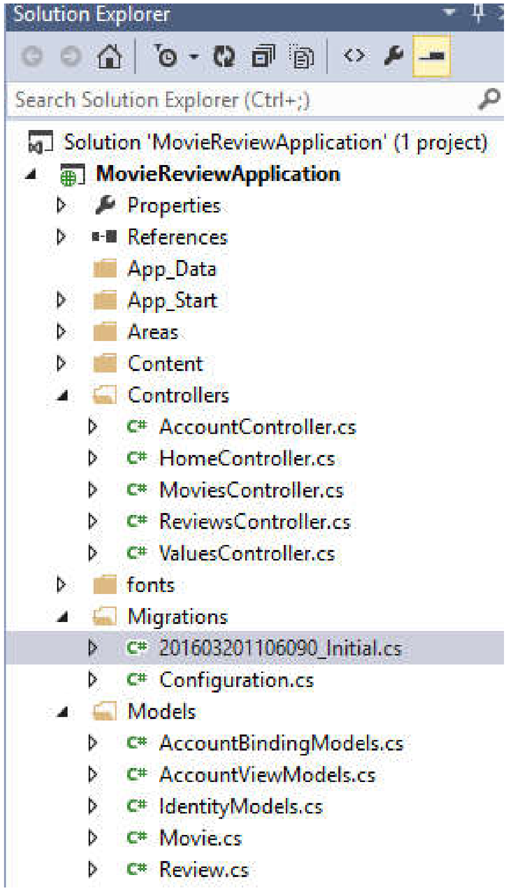
Model class: Review (MovieReviewApplication.Models)

Data context class: ApplicationDbContext (MovieReviewApplication.Models)

□ Use async controller actions

Controller name: ReviewsController

Package Manager Console



Solution Explorer

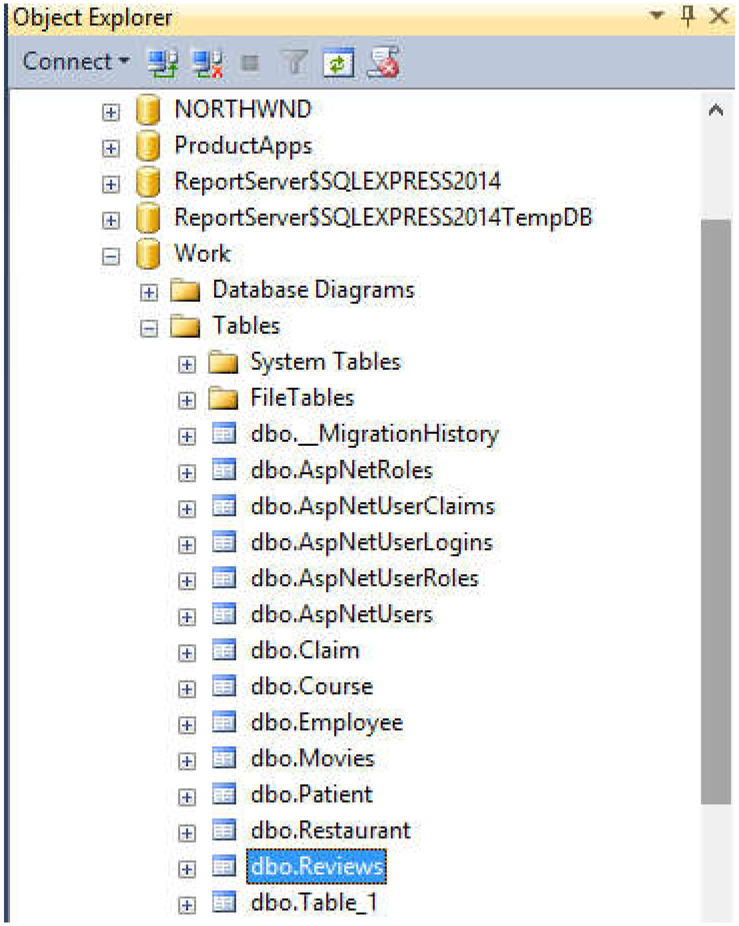
Search Solution Explorer (Ctrl+;)

Solution 'MovieReviewApplication' (1 project)
MovieReviewApplication
Properties
References
App_Data
App_Start
Areas
Content
Controllers
AccountController.cs
HomeController.cs
MoviesController.cs
ReviewsController.cs
ValuesController.cs
fonts
Migrations
201603201106090_Initial.cs
Configuration.cs
Models
AccountBindingModels.cs
AccountViewModels.cs
IdentityModels.cs
Movie.cs
Review.cs

```
Enable-Migrations
```

```
Add-Migration Initial
```

```
Update-Database
```

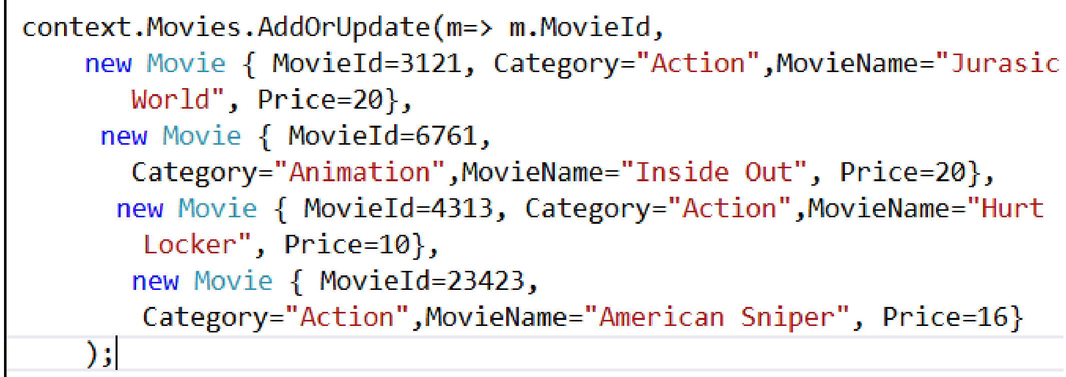


Object Explorer

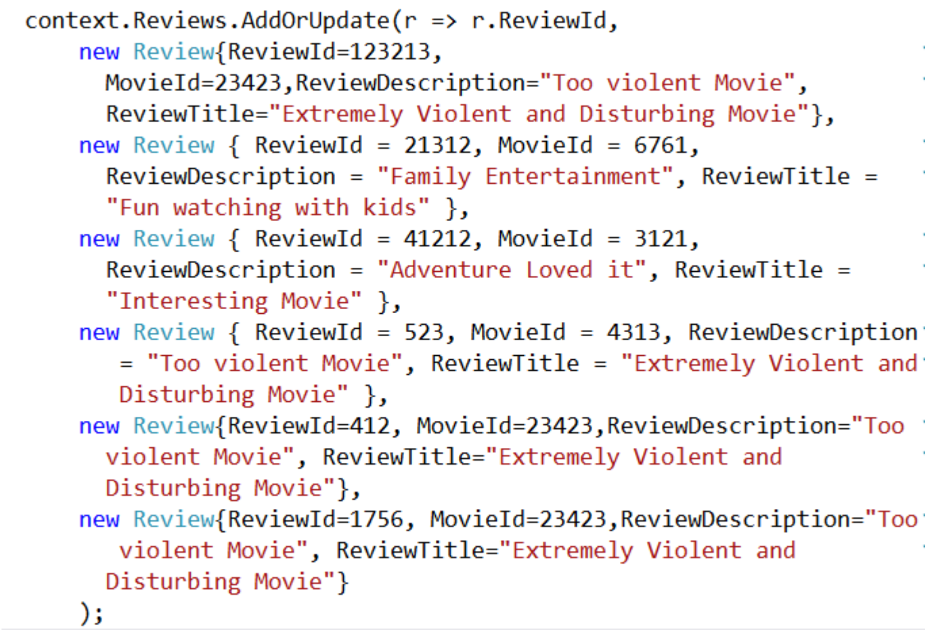
NORTHWND
ProductApps
ReportServer$SQLEXPRESS2014
ReportServer$SQLEXPRESS2014TempDB
Work
Database Diagrams
Tables
System Tables
FileTables
dbo.__MigrationHistory
dbo.AspNetRoles
dbo.AspNetUserClaims
dbo.AspNetUserLogins
dbo.AspNetUserRoles
dbo.AspNetUsers
dbo.Claim
dbo.Course
dbo.Employee
dbo.Movies
dbo.Patient
dbo.Restaurant
dbo.Reviews
dbo.Table_1

C#

Migrations -> Configuration.cs



```
context.Movies.AddOrUpdate(m=> m.MovieId,
    new Movie { MovieId=3121, Category="Action",MovieName="Jurasic
       World", Price=20},
     new Movie { MovieId=6761,
       Category="Animation",MovieName="Inside Out", Price=20},
      new Movie { MovieId=4313, Category="Action",MovieName="Hurt
        Locker", Price=10},
       new Movie { MovieId=23423,
        Category="Action",MovieName="American Sniper", Price=16}
    );
```

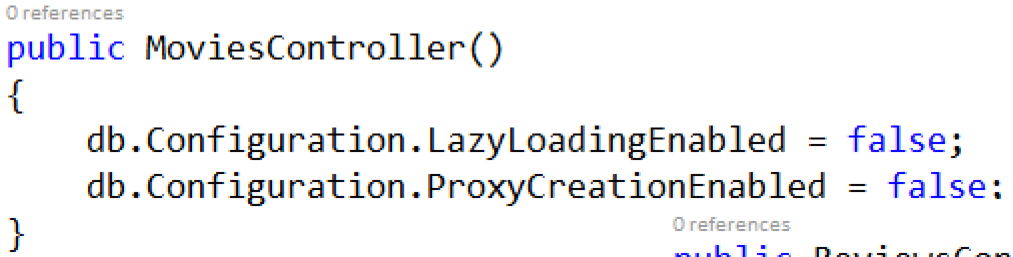


```
context.Reviews.AddOrUpdate(r => r.ReviewId,
    new Review{ReviewId=123213,
      MovieId=23423,ReviewDescription="Too violent Movie",
      ReviewTitle="Extremely Violent and Disturbing Movie"},
    new Review { ReviewId = 21312, MovieId = 6761,
      ReviewDescription = "Family Entertainment", ReviewTitle =
      "Fun watching with kids" },
    new Review { ReviewId = 41212, MovieId = 3121,
      ReviewDescription = "Adventure Loved it", ReviewTitle =
      "Interesting Movie" },
    new Review { ReviewId = 523, MovieId = 4313, ReviewDescription
       = "Too violent Movie", ReviewTitle = "Extremely Violent and
       Disturbing Movie" },
    new Review{ReviewId=412, MovieId=23423,ReviewDescription="Too
      violent Movie", ReviewTitle="Extremely Violent and
      Disturbing Movie"},
    new Review{ReviewId=1756, MovieId=23423,ReviewDescription="Too
       violent Movie", ReviewTitle="Extremely Violent and
      Disturbing Movie"}
    );
```

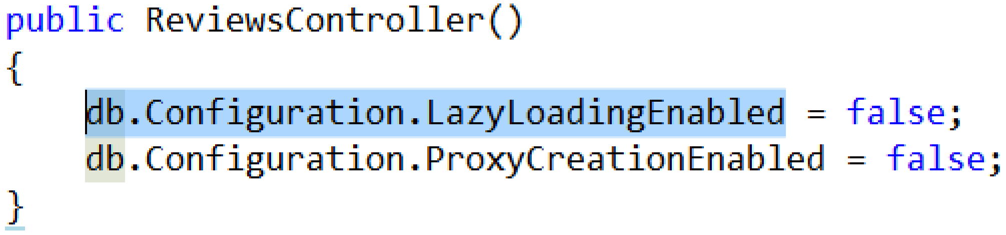
C# https://msdn.microsoft.com/en-in/data/jj713564.aspx

Relationship FixUP

- Navigation properties are not loaded explicitly
- Cyclic Reference

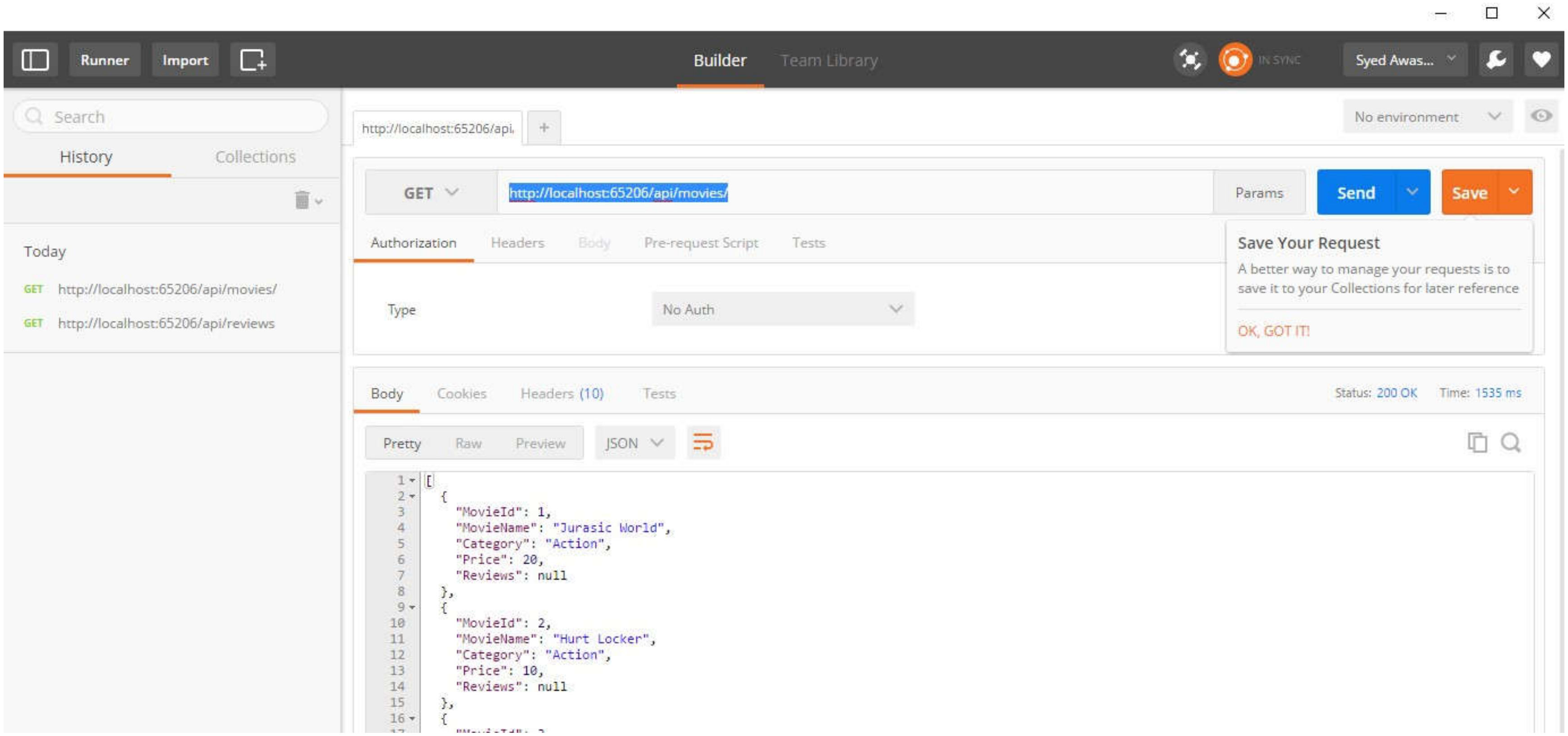


```
0 references
public MoviesController()
{
    db.Configuration.LazyLoadingEnabled = false;
    db.Configuration.ProxyCreationEnabled = false;
}
```



```
0 references
public ReviewsController()
{
    db.Configuration.LazyLoadingEnabled = false;
    db.Configuration.ProxyCreationEnabled = false;
}
```

Run the Application



Today

GET http://localhost:65206/api/movies/

GET http://localhost:65206/api/reviews

GET http://localhost:65206/api/movies/

Type: No Auth

Save Your Request

A better way to manage your requests is to save it to your Collections for later reference

OK, GOT IT!

Status: 200 OK Time: 1535 ms

```
[
  {
    "MovieId": 1,
    "MovieName": "Jurasic World",
    "Category": "Action",
    "Price": 20,
    "Reviews": null
  },
  {
    "MovieId": 2,
    "MovieName": "Hurt Locker",
    "Category": "Action",
    "Price": 10,
    "Reviews": null
  },
  {
```

SYED AWASE KHIRNI

MEDIATYPEFORMATTERS IN WEB API

SERIALIZATION

The process of translating a .NET Common Language Runtime (CLR) type into format that can be transmitted over HTTP. The default format can be either JSON or XML.

A media type formatter that is an object of type **MediaTypeFormatter** performs the serialization in the ASP.NET Web API pipeline.

MediaTypeFormatter

- To seamlessly convert HTTP data to/from .NET types.
- **media type:** refers to the value of the content-type header within an HTTP request and response.
  - Media types allow agent (client) and server to define the type of the data sent in the HTTP body (payload).
  - It is also used within the accept header in the request to allow content negotiation, i.e. client notifies the server of the media types it accepts/prefers.

Media type formatter is the bridge between the HTTP world of URI fragments, headers and body on one side, and the controller world of actions, parameters and return types.

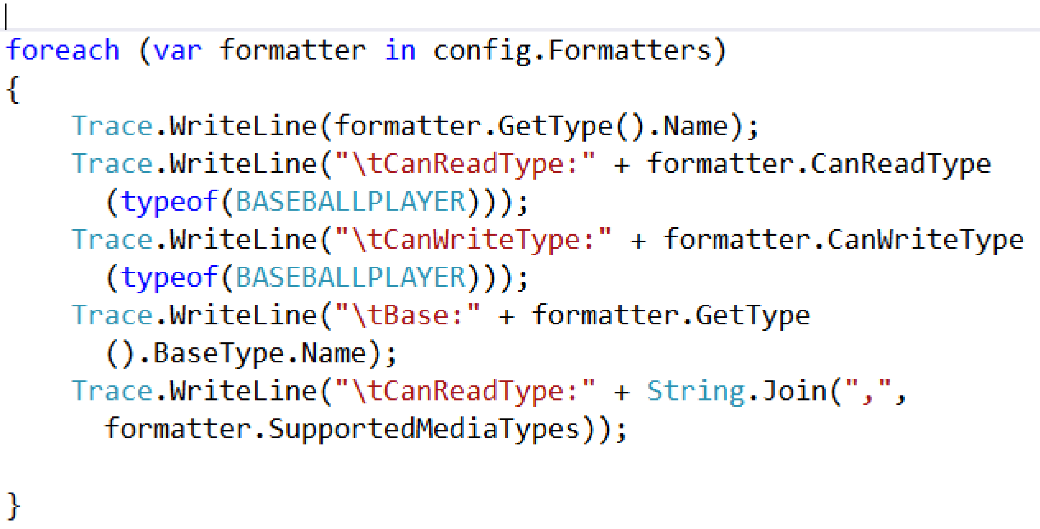
Using MediaTypeFormatters

- Global formatters sitting in the formatters property of **HttpConfiguration.**
- **If you are using ASP.NET hosting (IIS, Cassini, etc) then you can use GlobalConfiguration.Configuration to access the instance of HttpConfiguration containing formatters.**

App_Start-> WebApiConfig.cs

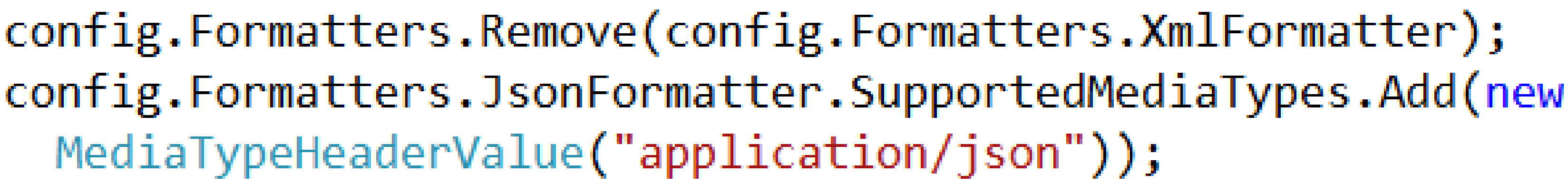
Result in Output window

- For Self-Hosting
  - Create a HttpSelfHostConfiguration object which has formatters property.



```
foreach (var formatter in config.Formatters)
{
    Trace.WriteLine(formatter.GetType().Name);
    Trace.WriteLine("\tCanReadType:" + formatter.CanReadType
      (typeof(BASEBALLPLAYER)));
    Trace.WriteLine("\tCanWriteType:" + formatter.CanWriteType
      (typeof(BASEBALLPLAYER)));
    Trace.WriteLine("\tBase:" + formatter.GetType
      ().BaseType.Name);
    Trace.WriteLine("\tCanReadType:" + String.Join(",",
      formatter.SupportedMediaTypes));

}
```

MediaTypeFormatters



```
config.Formatters.Remove(config.Formatters.XmlFormatter);
config.Formatters.JsonFormatter.SupportedMediaTypes.Add(new
  MediaTypeHeaderValue("application/json"));
```

Order of handling a request from a MediaType Formatter

- System.Net.Http.Formatting.JsonMediaTypeFormatter, based on JSON.NET
- System.Net.Http.Formatting.XmlMediaTypeFormatter, based on DataContractSerializer
- System.Net.Http.Formatting.FormUrlEncodedMediaTypeFormatter, for handling HTML form URL-encoded data
- System.Web.Http.ModelBinding.JQueryMvcFormUrlEncodedFormatter, for handling model-bound HTML form URL-encoded data