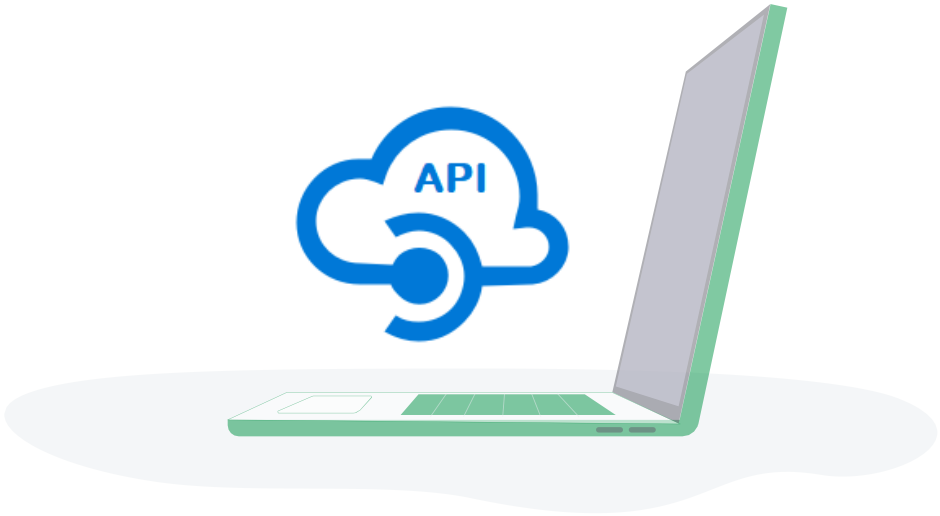
Web Application Programming Interface (API)

Tahaluf Training Center 2023



Overview Of ASP.NET Core

ASP.Net Core is the latest version of Microsoft's .NET Framework, which is a free, open-source, general-purpose development platform. It's a cross-platform framework that works with Windows, Mac OS X, and Linux.

Many applications can be made using ASP.NET Core, Such as Internet of Things (IoT) apps, web apps, and mobile backends. Can run on the cloud or on-premises.

Why ASP.NET Core ?

Why ASP.NET Core

- Cross-platform
- Open-source.
- Development methods (can develop on multiple platforms)
- lightweight framework.
- Deployment including Containerization
- A cloud-ready.
- Built-in dependency injection.
- High speed and performance

DI (Dependency Injection) is a design pattern for software. It enables us to write code that is loosely coupled. Dependency Injection's goal is to make code more manageable. Dependency Injection helps in the reduction of tight coupling between program components. Dependency Injection eliminates hard-coded dependencies between your classes by injecting them at runtime rather than at design time.

Differences between ASP.NET Core and ASP.NET

BASED ON	.NET Core	.NET Framework
Open Source	.Net Core is an open source.	The .Net Framework contains a few open source components.
Cross-Platform	(cross-platform) compatible with various operating systems — Windows, Linux, and Mac OS.	compatible with the only windows operating system.
Application Models	. Net Core does not support the development of desktop applications; instead, it is focused on the web, Windows Mobile, and the Windows Store.	. The Net Framework is used to create desktop and web applications, and it also supports WPF and Windows Forms applications.

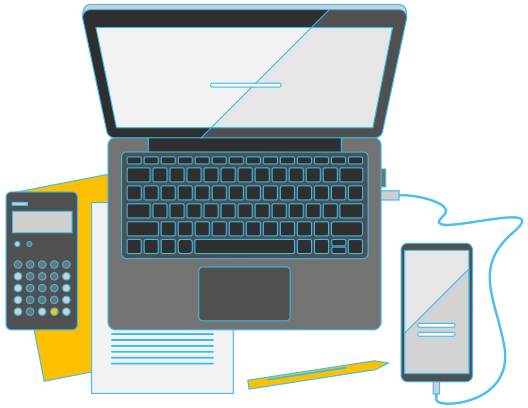
BASED ON	.NET Core	.NET Framework
Compatibility	.NET Core is compatible with various operating systems — Windows, Linux, and Mac OS.	.NET Framework is compatible only with the Windows operating system.
Packaging and Shipping	.Net Core software is distributed as a collection of Nugget packages.	The.Net Framework libraries are all packaged and provided as a single unit.

BASED ON	.NET Core	.NET Framework
Support for Micro-Services and API Services	Micro-services may be created and implemented using .Net Core, and a REST API is created in order to accomplish this.	While REST API services are supported by .Net Framework, microservice creation and implementation are not.
Performance and Scalability	High performance and scalability are advantages of.NET Core.	In terms of performance and application scalability,.Net Framework performs less effectively than.Net Core.

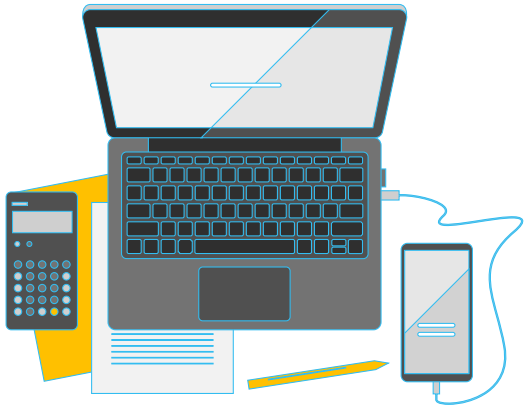
Difference between MVC & Web API

While **Asp.Net MVC** is used to build web applications that provide both views and data, **Asp.Net Web API** is used to quickly and easily create HTTP services that just return data.

Web API will also take care of returning data in a certain format, such as JSON, XML, or any other dependent on the Accept header in the request, so you don't have to bother about it. JsonResult is an **MVC** feature that exclusively returns data in JSON format.

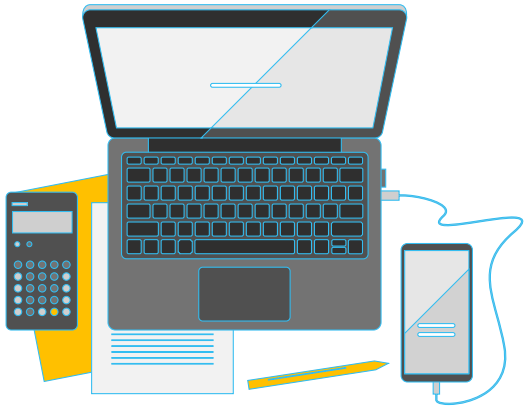


Use **ASP.Net MVC** if you want to provide services that are only relevant to one application. On the other hand, if your business requirements require you to offer the functionality generally, you would desire a **Web API** approach.



While the request is mapped to actions in **Web API** based on HTTP verbs, it is mapped to action names in **MVC**.

Additionally, **Web API** is a lightweight design that may be utilized with mobile apps in addition to web applications.



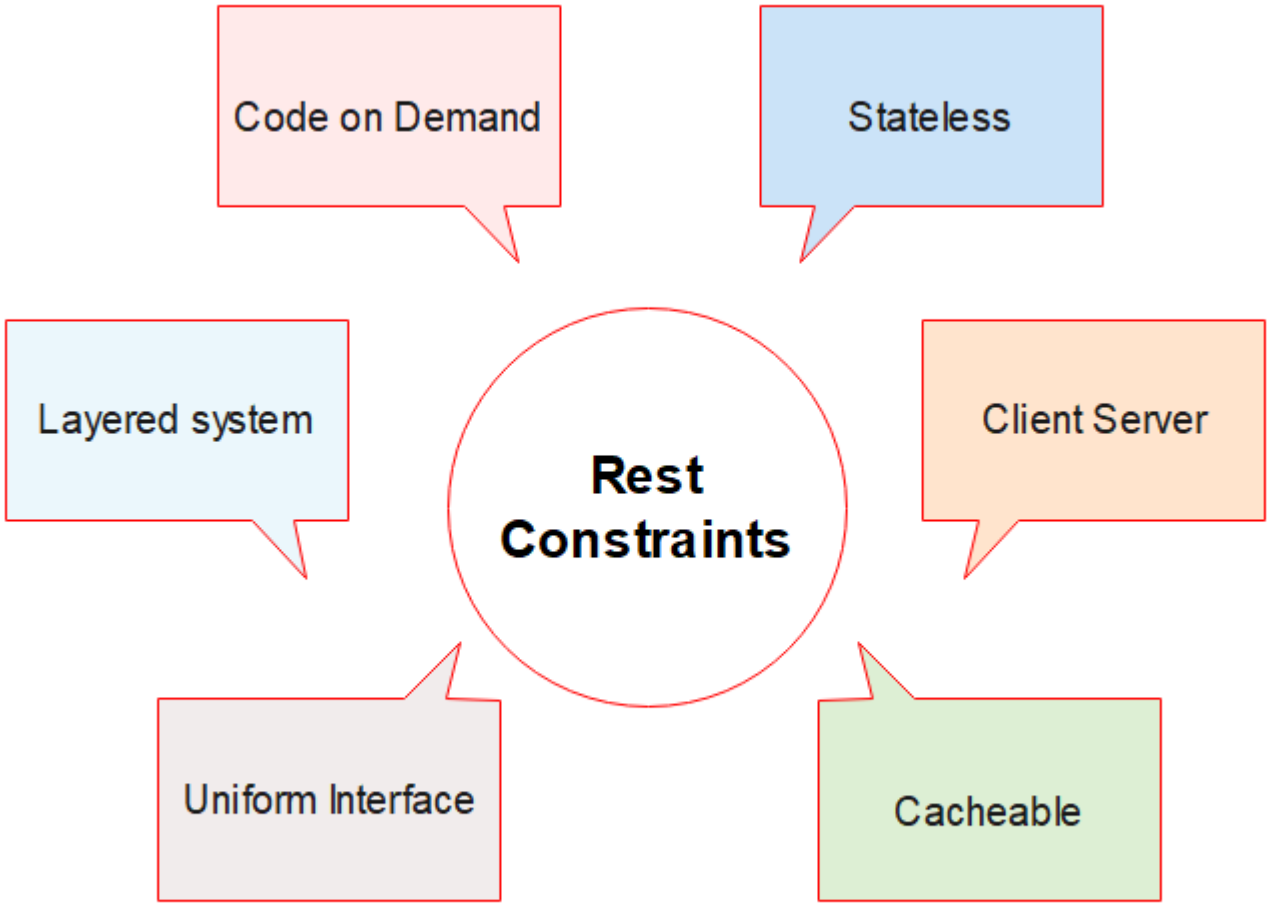
Overview of ASP.NET Web API

What are RESTful Services?

REST represents a Representational State Transfer.

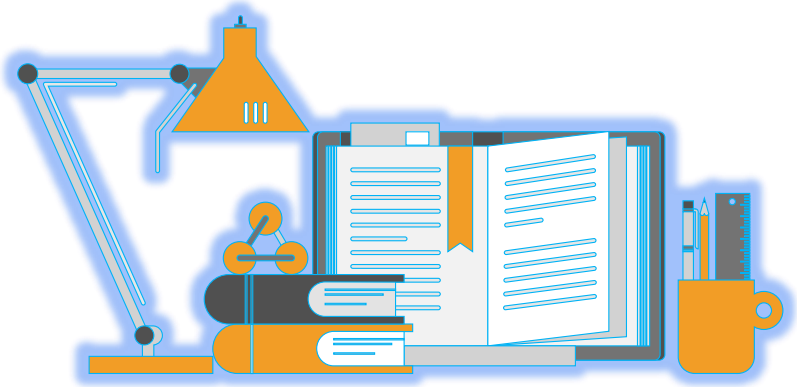
REST is an architectural pattern used to create an API that uses **HTTP** as a communication method.

REST specifies a collection of constraints that a system must adhere to.



❑ Client Server

A client sends a request, then a Server sends a response. This separation of concerns supports the independence and evolution of server-side logic and client-side logic.

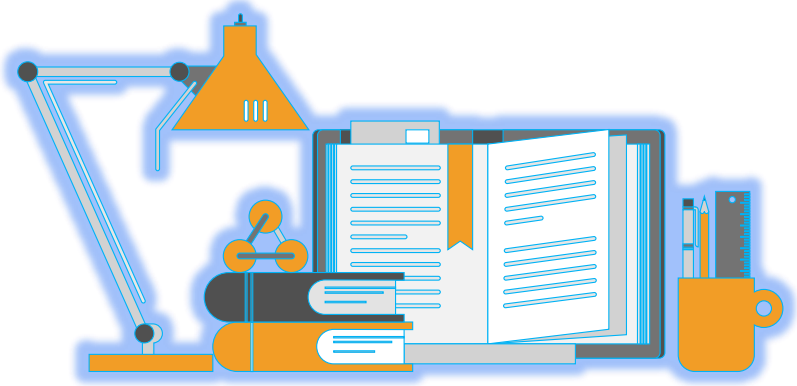


❑ Stateless

The communication between server and a client must be stateless.

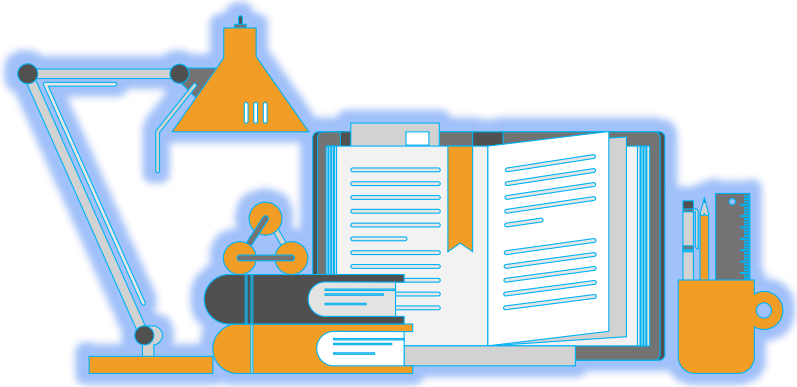
The request from the client should contain all the information for the server needs it to complete a process.

Each request is treated independently by the server.



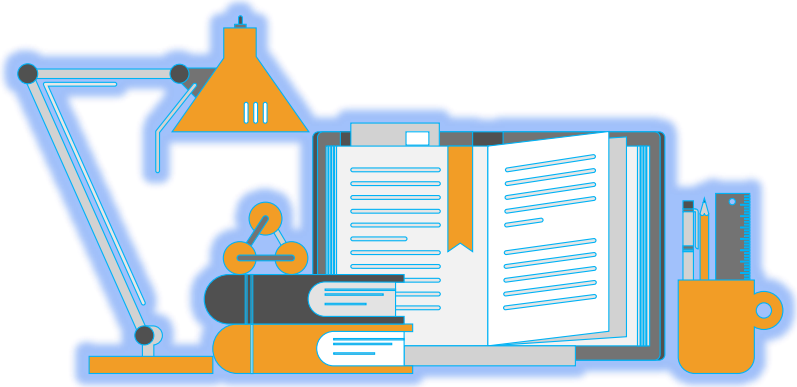
❑ Cacheable

Every response should specify whether or not it can be cached on the client side as well as how long it may be cached there. For any subsequent requests, the client will return the data from its cache, eliminating the need to resend the request to the server.



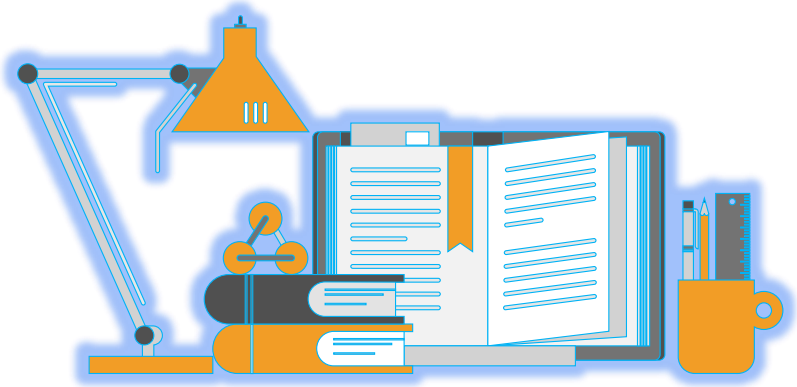
❑ Uniform Interface

It implies that there should be a standard way of interacting with a certain server regardless of the device or kind of application (website, mobile app).



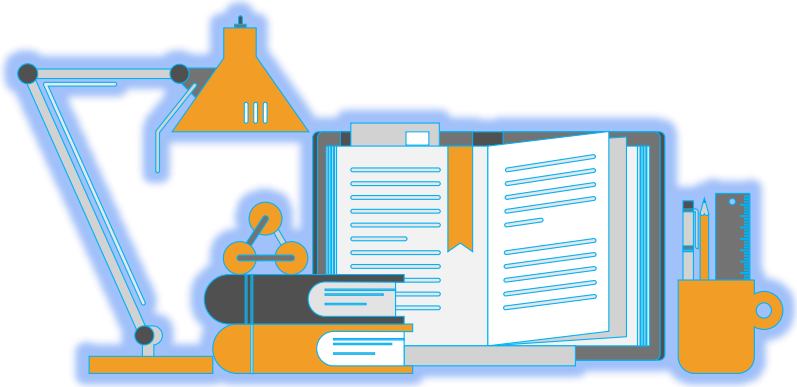
❑ Layered system

An application architecture must be composed of several levels. There are several intermediary servers between the client and the end server, and each layer has no knowledge of any layer other than its immediate layer.



❑ Code on demand

It is an optional feature. Servers can also give executable code to clients, according to this such as JavaScript code.



References

1. Complete Guide to Test Automation Arnon Axelrod 2018.

2. https://martinfowler.com/articles/is-tdd-dead/

3. https://seleniumhq.wordpress.com/2017/08/09/firefox-55-and-selenium-ide/

The Learning Hub

Thank You