# ASP.NET Core Web Api入门

## 课前准备

1. 学员基础：熟悉C#编程，了解ASP.NET
2. 演示环境：Windows 10 + Visual Studio 2022 + .NET 6
3. Visual Studio 2022：<https://visualstudio.microsoft.com/zh-hans/>
4. .NET 6 : <https://dotnet.microsoft.com/download>
5. 课程源码：<https://github.com/zmrbak/WebApi>
6. 参考资料：<https://docs.microsoft.com/zh-cn/aspnet/core/?view=aspnetcore-6.0>
7. 课程类别：演示、实操教学
8. 相关书籍：不做推荐，都挺好的，但是需要具备相应的知识储备。

## 第一个Web Api项目

curl -X "GET" "https://localhost:7037/WeatherForecast" -H "accept: text/plain"

Windows 10系统中运行前，把单引号换成双引号，删除\，将其写成一行。

Linux中，两种都可以。

## 将模板项目改造成实用的Web Api（1）

[HttpGet]

curl -X 'GET' \

'https://localhost:44319/WeatherForecast' \

-H 'accept: text/plain'

[HttpGet("{index}")]

curl -X 'GET' \

'https://localhost:44319/WeatherForecast/**1**' \

-H 'accept: text/plain'

[HttpPost]

public WeatherForecast PostWeatherForecast(**int index**)

{

return weatherForecasts.ElementAt(index);

}

curl -X 'POST' \

'https://localhost:44319/WeatherFor**ecast?in**dex=1' \

-H 'accept: text/plain' \

-d ''

1. Microsoft.AspNetCore.Mvc.HttpDeleteAttribute
2. Microsoft.AspNetCore.Mvc.HttpGetAttribute
3. Microsoft.AspNetCore.Mvc.HttpHeadAttribute
4. Microsoft.AspNetCore.Mvc.HttpOptionsAttribute
5. Microsoft.AspNetCore.Mvc.HttpPatchAttribute
6. Microsoft.AspNetCore.Mvc.HttpPostAttribute
7. Microsoft.AspNetCore.Mvc.HttpPutAttribute

GET：获取资源，安全，幂等

POST：创建资源，不安全，非幂等

PUT：更新/创建特定资源（替换一个资源），不安全，幂等

**DELETE：删除特定资源，不安全，幂等**

PATCH：对资源进行部分更新（修改某个资源中的一部分，打补丁），不安全，幂等

HEAD：与GET相同，**但不返回Body**，安全，幂等

OPTIONS：获取指定资源所支持的操作，安全，幂等

## 将模板项目改造成实用的Web Api（2）

## 将模板项目改造成实用的Web Api（3）

## 将Web Api发布到IIS服务器

1、安装IIS，使用默认配置

2、ASP.NET Core Runtime Hosting Bundle

3、将发布的Web Api复制到Web根目录

4、测试

curl -X "GET" ^

"http://localhost/WeatherForecast" ^

-H "accept: text/plain"

curl http://localhost/WeatherForecast

## 在IIS服务器上启用HTTPS(1)

Programe.cs

app.UseHttpsRedirection();

launchSettings.json

"sslPort": 44396

"applicationUrl": <https://localhost:7291;http://localhost:5291>

## 在IIS服务器上启用HTTPS(2)

## 将Web Api发布到Linux服务器

**安装aspnetcore运行时**

dotnet

yum makecache

yum search dotnet

yum search aspnetcore

yum install -y aspnetcore-runtime-6.0

dotnet

dotnet --info

**复制文件**

**运行程序**

cd publish/

dotnet WebApi9.dll

6.0.0-rc.2.21470.23

**修改配置**

vi WebApi9.runtimeconfig.json

6.0.0-rc.2.21470.23

**运行程序**

dotnet WebApi9.dll

**测试**

curl http://localhost:5000/WeatherForecast/

ip a

curl <http://192.168.240.134:5000/WeatherForecast/>

**设置端口**

dotnet WebApi 9.dll --urls="http://\*:5001"

curl <http://192.168.240.134:5001/WeatherForecast/>

**外部访问**

curl <http://192.168.240.134:5001/WeatherForecast/>

systemctl stop firewalld

curl <http://192.168.240.134:5001/WeatherForecast/>

systemctl start firewalld

curl <http://192.168.240.134:5001/WeatherForecast/>

## 使用Nginx来发布Web Api

Kestrel

Nginx

**安装nginx代理**

yum install -y nginx

systemctl enable nginx

/usr/lib/systemd/system/nginx.service

systemctl start nginx

**配置nginx**

**https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/linux-nginx?view=aspnetcore-6.0**

vi /etc/nginx/nginx.conf

location / {

proxy\_pass http://127.0.0.1:5000;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection keep-alive;

proxy\_set\_header Host $host;

proxy\_cache\_bypass $http\_upgrade;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}

nginx -s reload

**添加防火墙策略**

firewall-cmd --add-port=80/tcp --permanent

firewall-cmd --reload

**SELinux设置**

setenforce 0

curl http://192.168.240.134/WeatherForecast/

sestatus

setenforce 1

**SELinux排错工具**

yum install -y setroubleshoot

cat /var/log/audit/audit.log

tail -n 3 /var/log/audit/audit.log

tail -n 3 /var/log/audit/audit.log | tee /tmp/audit

audit2why -i /tmp/audit

setsebool -P httpd\_can\_network\_connect 1

## 将Web Api以Linux中的服务来运行

**创建服务配置文件**

vi /etc/systemd/system/kestrel-WebApi009.service

**添加内容**

[Unit]

Description=Example .NET Web API App running on Ubuntu

[Service]

WorkingDirectory=/root/publish/

ExecStart=/usr/bin/dotnet /root/publish/WebApi009.dll

Restart=always

# Restart service after 10 seconds if the dotnet service crashes:

RestartSec=10

KillSignal=SIGINT

SyslogIdentifier=dotnet-example

User=root

Environment=ASPNETCORE\_ENVIRONMENT=Production

Environment=DOTNET\_PRINT\_TELEMETRY\_MESSAGE=false

[Install]

WantedBy=multi-user.target

**启用服务**

systemctl enable kestrel-WebApi009

**启动服务**

systemctl start kestrel-WebApi009

**参考文档**

https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/linux-nginx?view=aspnetcore-6.0

## Web Api中的Route

## Web Api中的Action以及数据绑定