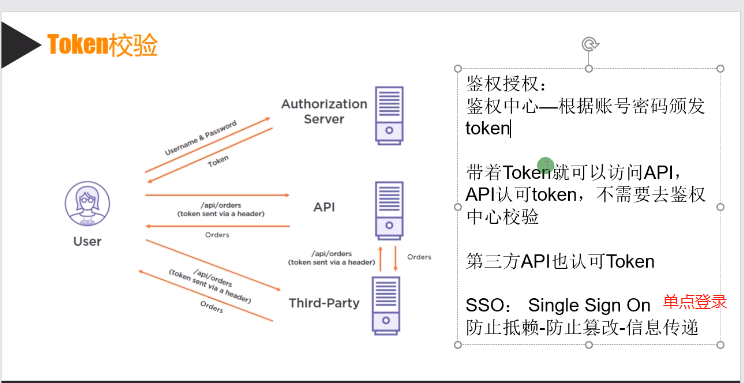
推荐使用 IdentityServer4

Token:官网 jwt.io

## JWT





## 使用：

1. 引入System.IdentityModel.Tokens.Jwt，Microsoft.AspNetCore.Authentication.JwtBearer包
2. 使用

[HttpPost]

[Route("login")]

[Authorize]

public string Login(string name, string password)

{

if ("zy".Equals(name) && "123456".Equals(password))

{

//账号密码正确，获取token

string token = \_jWTService.GetToken(name);

return JsonConvert.SerializeObject(new

{

result = true,

token

});

}

else {

return JsonConvert.SerializeObject(new

{

result = false,

token=""

});

}

}

1. Startup. ConfigureServices

//注入

services.AddScoped<IJWTService, JWTService>();

#region jwt鉴权授权

var audience = this.Configuration["audience"];

var issuer = this.Configuration["issuer"];

var SecurityKey = this.Configuration["SecurityKey"];

services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)//默认授权机制

.AddJwtBearer(options=> {

options.TokenValidationParameters = new Microsoft.IdentityModel.Tokens.TokenValidationParameters

{

ValidateIssuer = true,//是否验证

ValidateAudience=true,

ValidateLifetime=true,

ValidateIssuerSigningKey=true,

ValidAudience= audience,

ValidIssuer= issuer,

IssuerSigningKey=new SymmetricSecurityKey(Encoding.UTF8.GetBytes(SecurityKey))

};

});

1. Startup. Configure

//通过中间件来进行鉴权授权

app.UseAuthentication();

1. appsettings.json

"audience": "http://localhost:10086",

"issuer": "http://localhost:10086",

"SecurityKey": ""

1. 创建授权类JWTService

public interface IJWTService

{

string GetToken(string username);

}

public class JWTService : IJWTService

{

private readonly IConfiguration \_iConfiguration;

public JWTService(IConfiguration iConfiguration)

{

\_iConfiguration = iConfiguration;

}

public string GetToken(string username)

{

/\*\*

1. \* claims Payload 有效载荷，包含了一些和token有关的信息，建议配置在 配置文件appsettings.json

中

\* 这些字段是关键字，除了这些可以包含其他任何json可以兼容的字段

\* issuer：给谁的

\* sub:主题

\* iat：时间戳

\* jti: token唯一标识

\* audience:签发人

\*/

Claim[] claims = new[] {

new Claim(ClaimTypes.Name,username),

new Claim("name", "zy"),

new Claim("age", "18"),

new Claim("money","5000000") //其他字段信息

};

SymmetricSecurityKey key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_iConfiguration["SecurityKey"]));

SigningCredentials creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

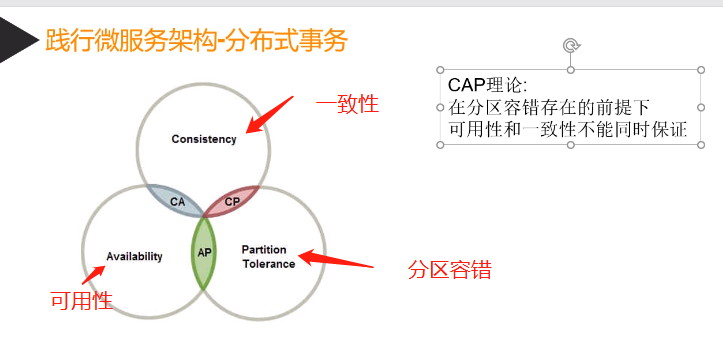
var token = new JwtSecurityToken(issuer: \_iConfiguration["issuer"],audience: \_iConfiguration["audience"],claims: claims,expires:DateTime.Now.AddMinutes(5),signingCredentials: creds);

string returntoken = new JwtSecurityTokenHandler().WriteToken(token);

return returntoken;

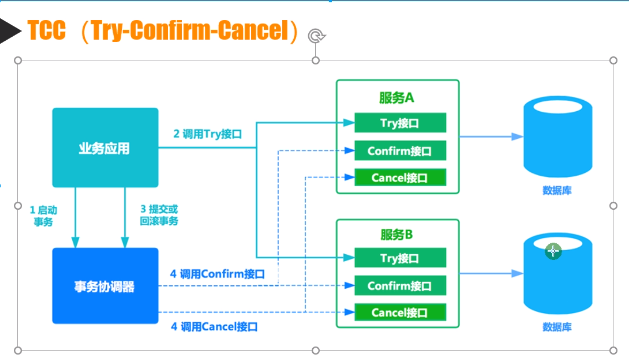
}

}

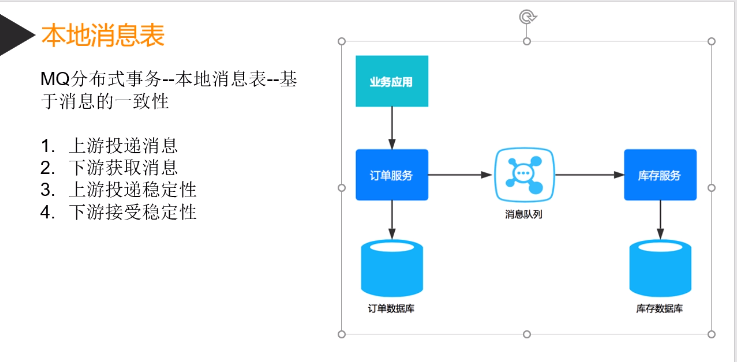
2pc/3pc

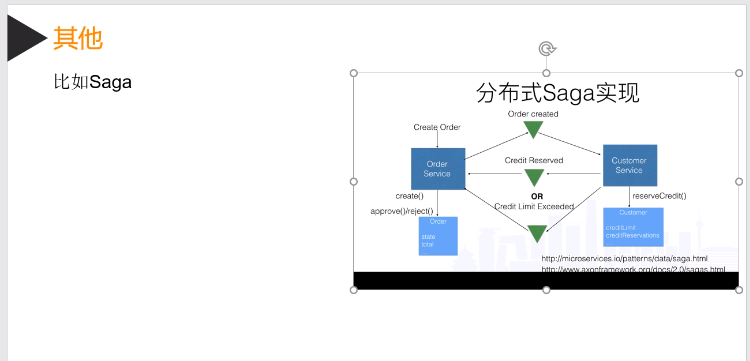
Base理论：分区容错前提下，优先保证可用性，然后最终结果一致就行。

Tcc: 银行用的多



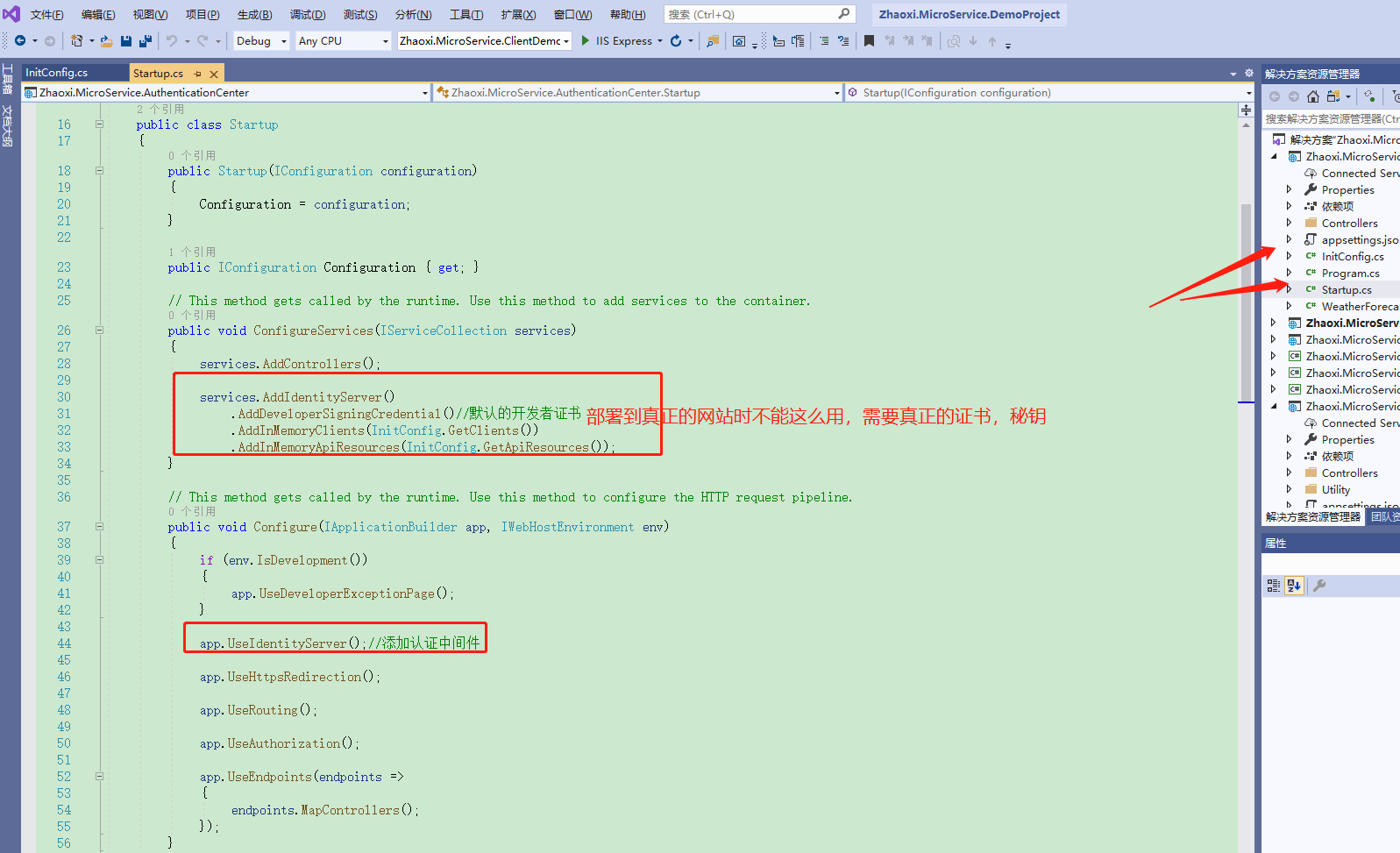
互联网常用:



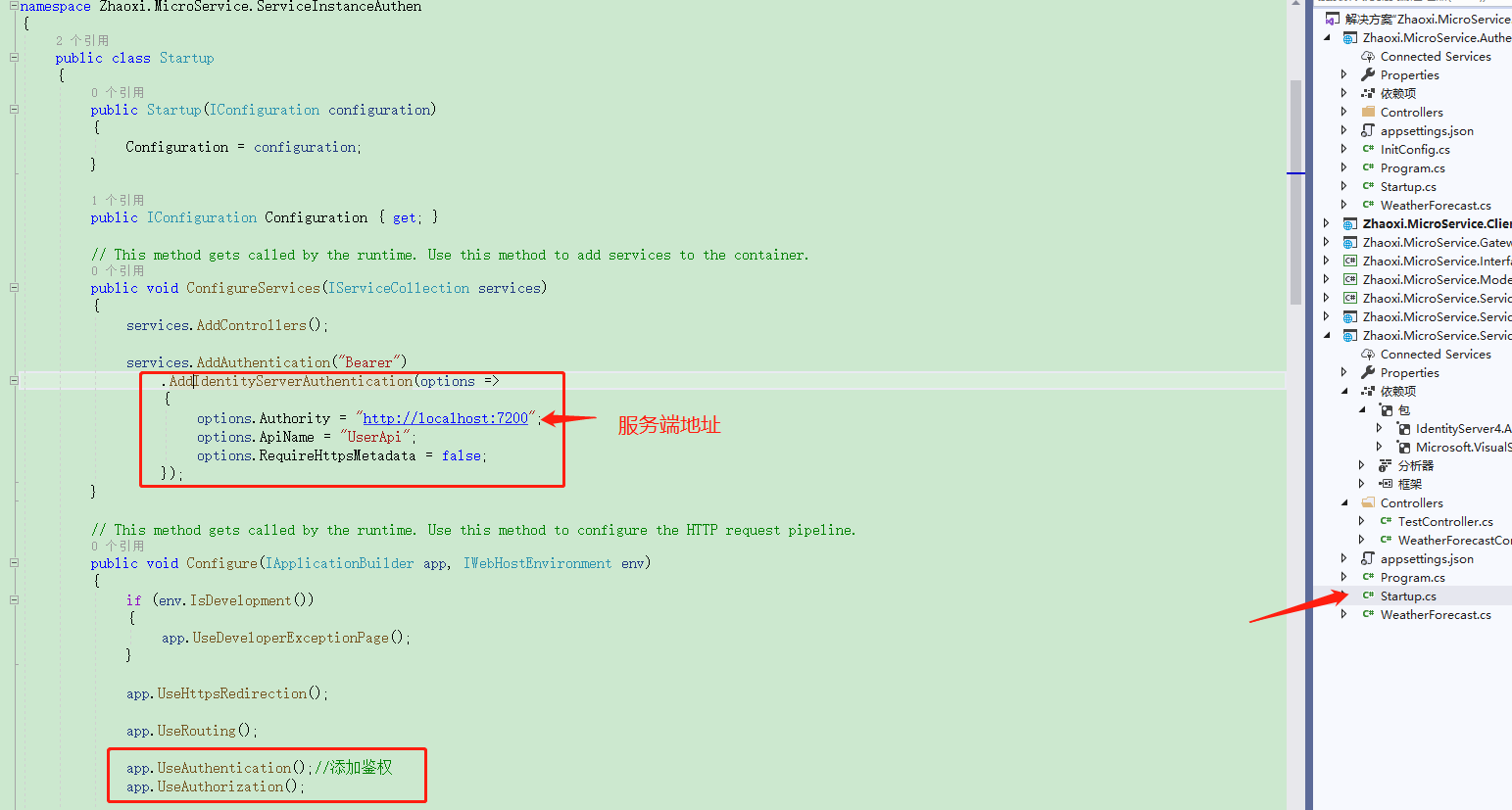


## IdentityServer4

添加IdentityServer4包 (服务端)



客户端 添加IdentityServer4.AccessTokenValidation 包：



然后在需要的地方添加授权特性[Authorize] ，没有授权，返回401

[Authorize]

[HttpGet]

[Route("IndexA")]

public IActionResult IndexA()

{

return new JsonResult(new

{

message = "This is TestControllerIndex",

Port = this.\_iConfiguration["port"],

Time = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss fff")

});

}