这个快速入门将展示如何构建一个JavaScript客户端应用程序。 用户将登录到IdentityServer，使用IdentityServer发出的访问令牌调用Web API，并注销IdentityServer。

**认证服务器配置**

# **为JavaScript客户端添加客户端注册到IdentityServer**

// JavaScript Client

new Client

{

ClientId = "js",

ClientName = "JavaScript Client",

AllowedGrantTypes = GrantTypes.Implicit,

AllowAccessTokensViaBrowser = true,

RedirectUris = { "http://localhost:5003/callback.html" },

PostLogoutRedirectUris = { "http://localhost:5003/index.html" },

AllowedCorsOrigins = { "http://localhost:5003" },

AllowedScopes =

{

IdentityServerConstants.StandardScopes.OpenId,

IdentityServerConstants.StandardScopes.Profile,

"api1"

}

}

**Api资源服务器配置**

# **为webapi添加CORS，允许跨域调用api**

最后一个必要的配置是在Web API项目中配置CORS。 这将允许Ajax调用从http://localhost:5003调用http://localhost:5001的webapi。

**配置CORS**

在Startup.cs的ConfigureServices中的将将CORS服务添加到依赖注入系统：

public void ConfigureServices(IServiceCollection services){

services.AddMvcCore()

.AddAuthorization()

.AddJsonFormatters();

services.AddAuthentication("Bearer")

.AddIdentityServerAuthentication(options =>

{

options.Authority = "http://localhost:5000";

options.RequireHttpsMetadata = false;

options.ApiName = "api1";

});

services.AddCors(options =>

{

// this defines a CORS policy called "default"

options.AddPolicy("default", policy =>

{

policy.WithOrigins("http://localhost:5003")

.AllowAnyHeader()

.AllowAnyMethod();

});

});

}

将CORS中间件添加到Configure中的管道：

public void Configure(IApplicationBuilder app)

{

app.UseCors("default");

app.UseAuthentication();

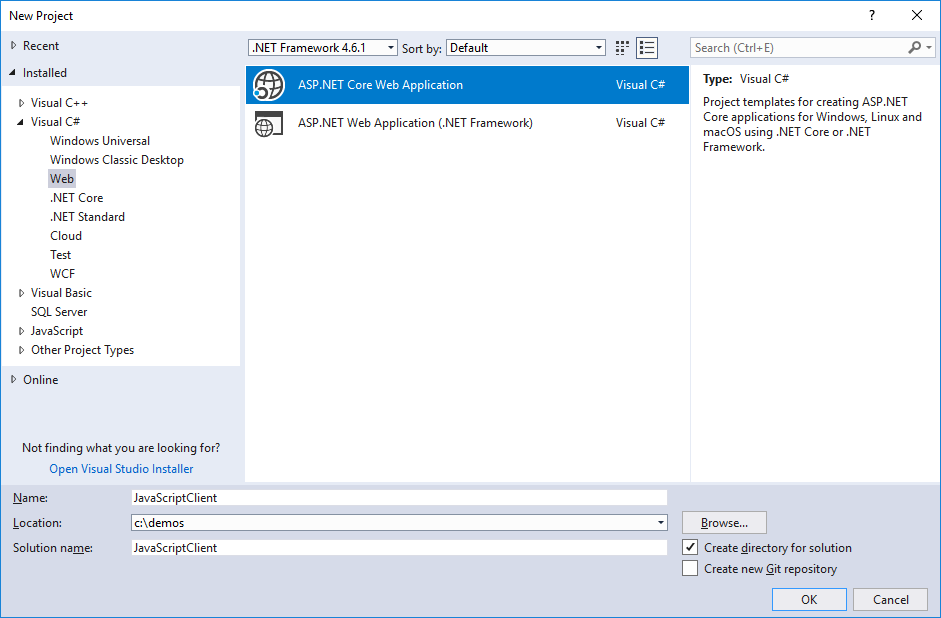
app.UseMvc();

}

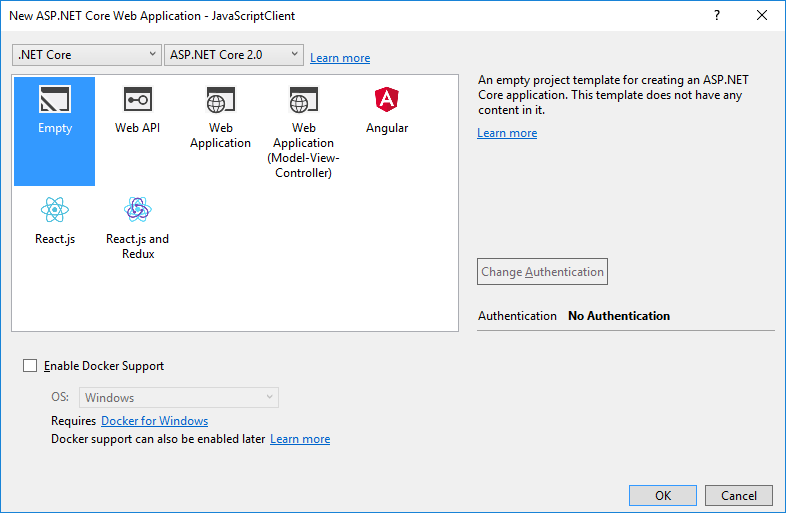
**新建客户端**

# **创建一个JavaScript客户端新项目**

创建一个新的Asp.NET Core web应用程序：



选择“空”模版：



# **修改端口**

将端口修改为5003

# **添加静态文件中间件**

在Startup.cs中的Configure方法中注册静态文件中间件：

public void Configure(IApplicationBuilder app)

{

app.UseDefaultFiles();

app.UseStaticFiles();

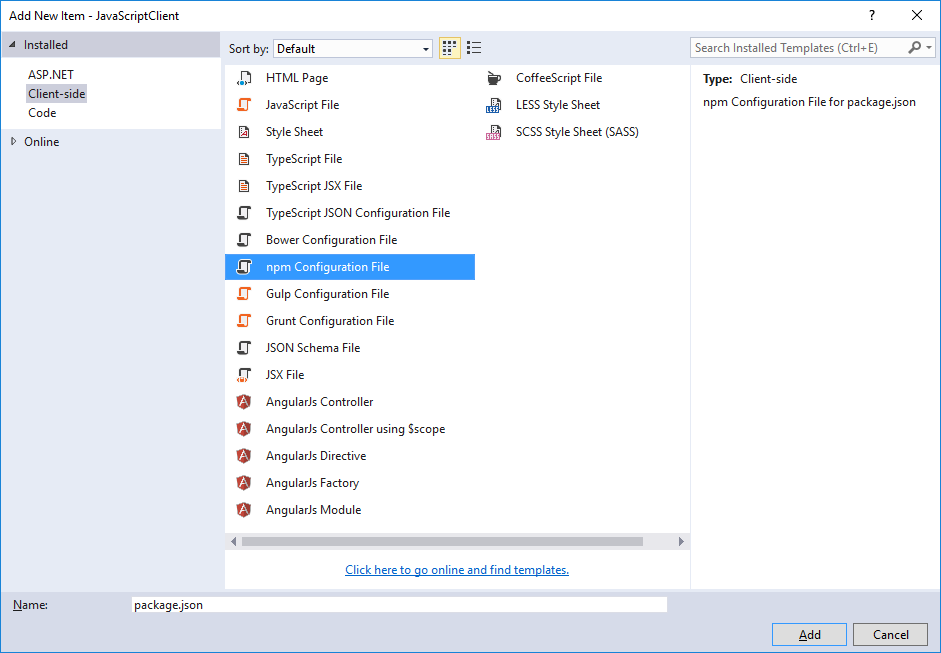
}

# **引用oidc客户端**

**NPM**

如果你需要使用NPM下载oidc-client库，你需要遵循下面的步骤：

添加一个NPM的包文件"package.json"



在package.json中添加oidc-client到devDependency：

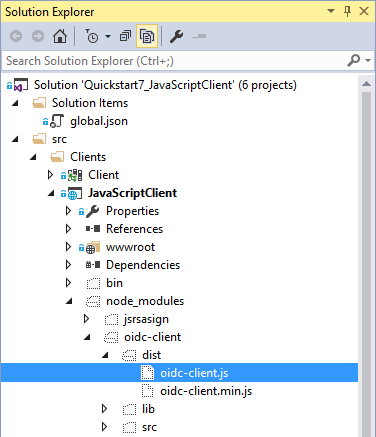
"devDependencies":

{

"oidc-client": "1.4.1"

}

保存此文件后，Visual Studio应自动将这些软件包恢复到名为node\_modules的文件夹中：



在〜/ node\_modules / oidc-client / dist文件夹中找到名为oidc-client.js的文件，并将其复制到应用程序的〜/ wwwroot文件夹中。

# **添加HTML和JavaScript文件**

接下来是将你的HTML和JavaScript文件添加到〜/ wwwroot。 我们将添加两个HTML文件和一个特定于应用程序的JavaScript文件（除了oidc-client.js库）。

**index.html**

这将是我们应用程序的主页。

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title></title></head><body>

<button id="login">Login</button>

<button id="api">Call API</button>

<button id="logout">Logout</button>

<pre id="results"></pre>

<script src="oidc-client.js"></script>

<script src="app.js"></script></body></html>

**app.js**

这将包含我们的应用程序的主要代码。 首先要添加一个辅助函数来将消息记录到

function log() {

document.getElementById('results').innerText = '';

Array.prototype.forEach.call(arguments, function (msg) {

if (msg instanceof Error) {

msg = "Error: " + msg.message;

}

else if (typeof msg !== 'string') {

msg = JSON.stringify(msg, null, 2);

}

document.getElementById('results').innerHTML += msg + '\r\n';

});}

接下来，添加代码注册三个按钮的“click”事件处理程序：

document.getElementById("login").addEventListener("click", login, false);document.getElementById("api").addEventListener("click", api, false);document.getElementById("logout").addEventListener("click", logout, false);

接下来，我们可以使用oidc-client库中的UserManager类来管理OpenID Connect协议。

var config = {

authority: "http://localhost:5000",

client\_id: "js",

redirect\_uri: "http://localhost:5003/callback.html",

response\_type: "id\_token token",

scope:"openid profile api1",

post\_logout\_redirect\_uri : "http://localhost:5003/index.html",

};

var mgr = new Oidc.UserManager(config);

接下来，UserManager提供一个getUser API来知道用户是否登录到JavaScript应用程序。

mgr.getUser().then(function (user) {

if (user) {

log("User logged in", user.profile);

}

else {

log("User not logged in");

}});

接下来，我们要实现登录，访问api和注销功能。 UserManager提供signinRedirect来登录用户，signoutRedirect登出用户。 添加此代码以在我们的应用程序中实现这三个函数：

function login() {

mgr.signinRedirect();

}

function api() {

mgr.getUser().then(function (user) {

var url = "http://localhost:5001/identity";

var xhr = new XMLHttpRequest();

xhr.open("GET", url);

xhr.onload = function () {

log(xhr.status, JSON.parse(xhr.responseText));

}

xhr.setRequestHeader("Authorization", "Bearer " + user.access\_token);

xhr.send();

});}

function logout() {

mgr.signoutRedirect();

}

**callback.html**

登录完成后，我们可以将用户重定向到主index.html页面。 添加此代码以完成登录过程：

<!DOCTYPE html><html><head>

<meta charset="utf-8" /> <title></title></head><body> <script src="oidc-client.js"></script>

<script>

new Oidc.UserManager().signinRedirectCallback().then(function () {

window.location = "index.html";

}).catch(function (e) {

console.error(e);

});

</script></body></html>

# **运行JavaScript应用**

现在你应该可以运行JavaScript客户端应用程序了：