IdentityServer是中间件和服务的组合。所有配置都在您的启动类中完成。

## **配置服务**

您可以通过调用以下方法将IdentityServer服务添加到DI系统：

**public** **void** ConfigureServices(IServiceCollection services){

var builder = services.AddIdentityServer();

}

这将返回一个构建器对象，该构建器对象又有许多方便的方法来连接其他服务。

## **密钥材料**

**AddSigningCredential**

添加签名密钥服务，该服务为各种令牌创建/验证服务提供指定的密钥材料。您可以从证书存储中传入证书的a X509Certificate2，a SigningCredential或引用。

**AddDeveloperSigningCredential**

在启动时创建临时密钥材料。当您没有要使用的证书时，这仅适用于dev。生成的密钥将保留到文件系统，以便在服务器重新启动之间保持稳定（可以通过传递禁用false）。

**AddValidationKey**

添加用于验证令牌的密钥。它们将由内部令牌验证器使用，并将显示在发现文档中。

## **内存配置存储**

使用这些配置API的目的是在原型设计，开发和/或测试时使用，在这种情况下，不需要在运行时为配置数据动态查询数据库。

**AddInMemoryClients**

基于配置对象的内存中集合的注册IClientStore和ICorsPolicyService实现Client。

**AddInMemoryIdentityResources**

IResourceStore根据IdentityResource配置对象的内存中集合注册实现。

**AddInMemoryApiResources**

IResourceStore根据ApiResource配置对象的内存中集合注册实现。

## **测试商店**

使用TestUser“内存”商店是因为它适用于原型设计，开发和/或测试。采用TestUser在生产中不推荐使用。

**AddTestUsers**

TestUserStore基于TestUser对象集合的注册。 TestUserStore由默认的快速入门UI使用。还注册IProfileService和的实现IResourceOwnerPasswordValidator。

## **配置管道**

您需要通过调用以下方法将IdentityServer添加到管道：

**public** **void** Configure(IApplicationBuilder app){

app.UseIdentityServer();}