Adding a UI

Gill Cleeren
CTO XPIRIT BELGIUM

@gillcleeren www.snowball.be

Overview

Introducing Blazor client-side

Using NSwag and NSwagStudio

Exploring the client app

Adding a missing feature

Introducing Blazor Client-side

Blazor is a framework
to build interactive web UIs
using C# and HTML.

Hello Blazor

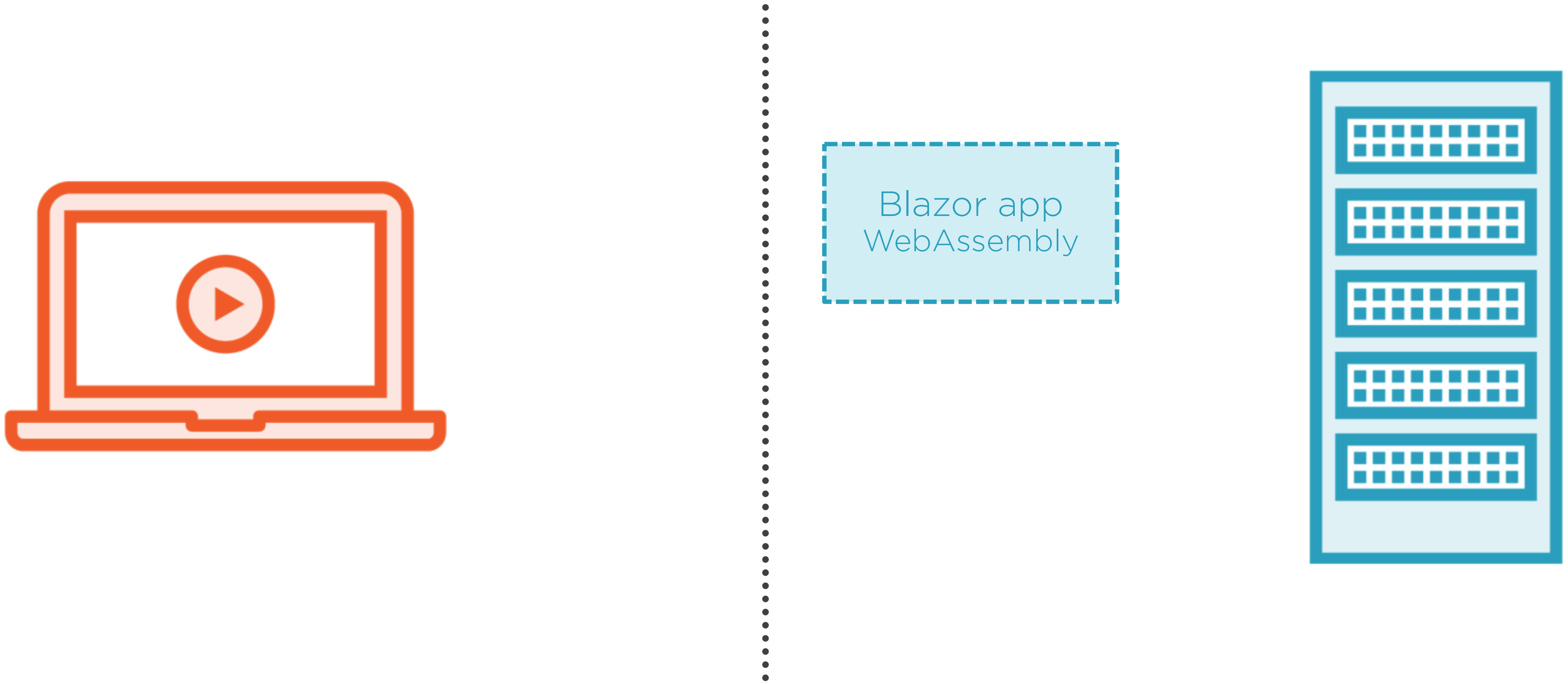
Based on WebAssembly or run on server

No plugin, based on web standards

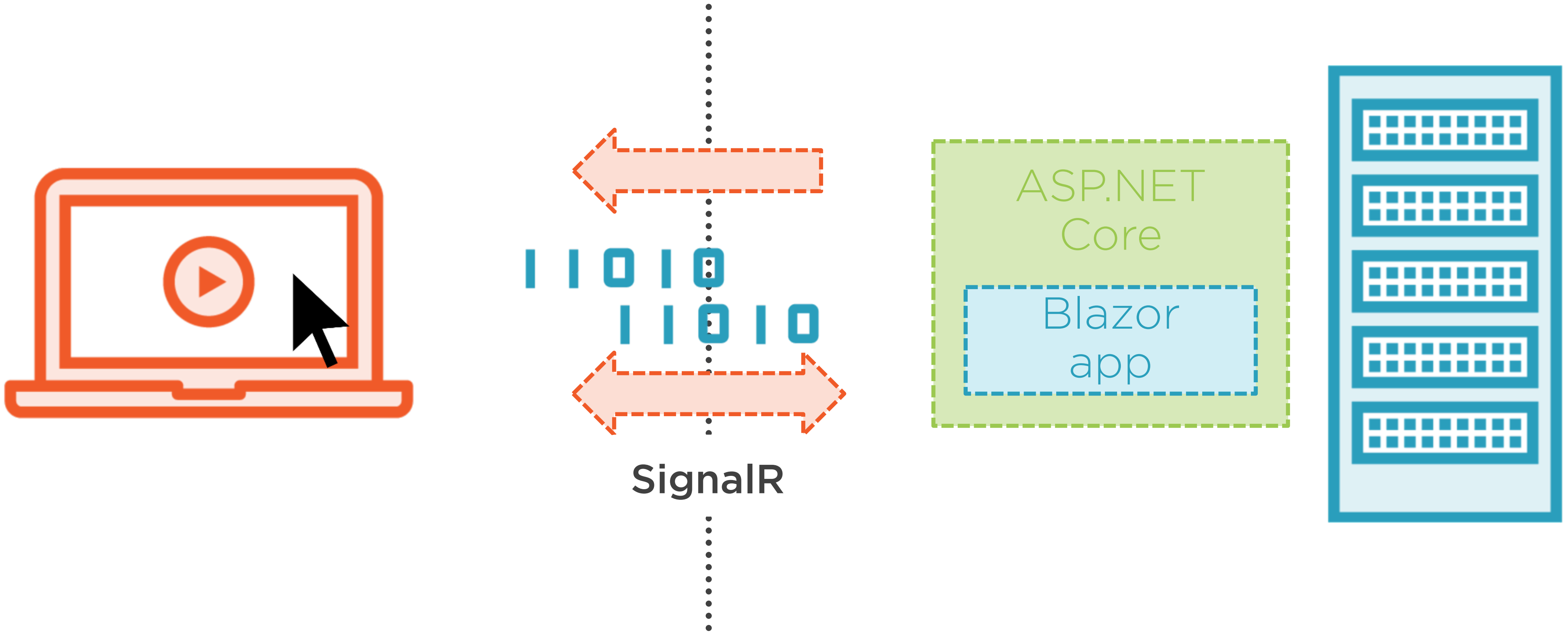
Integrate with JavaScript

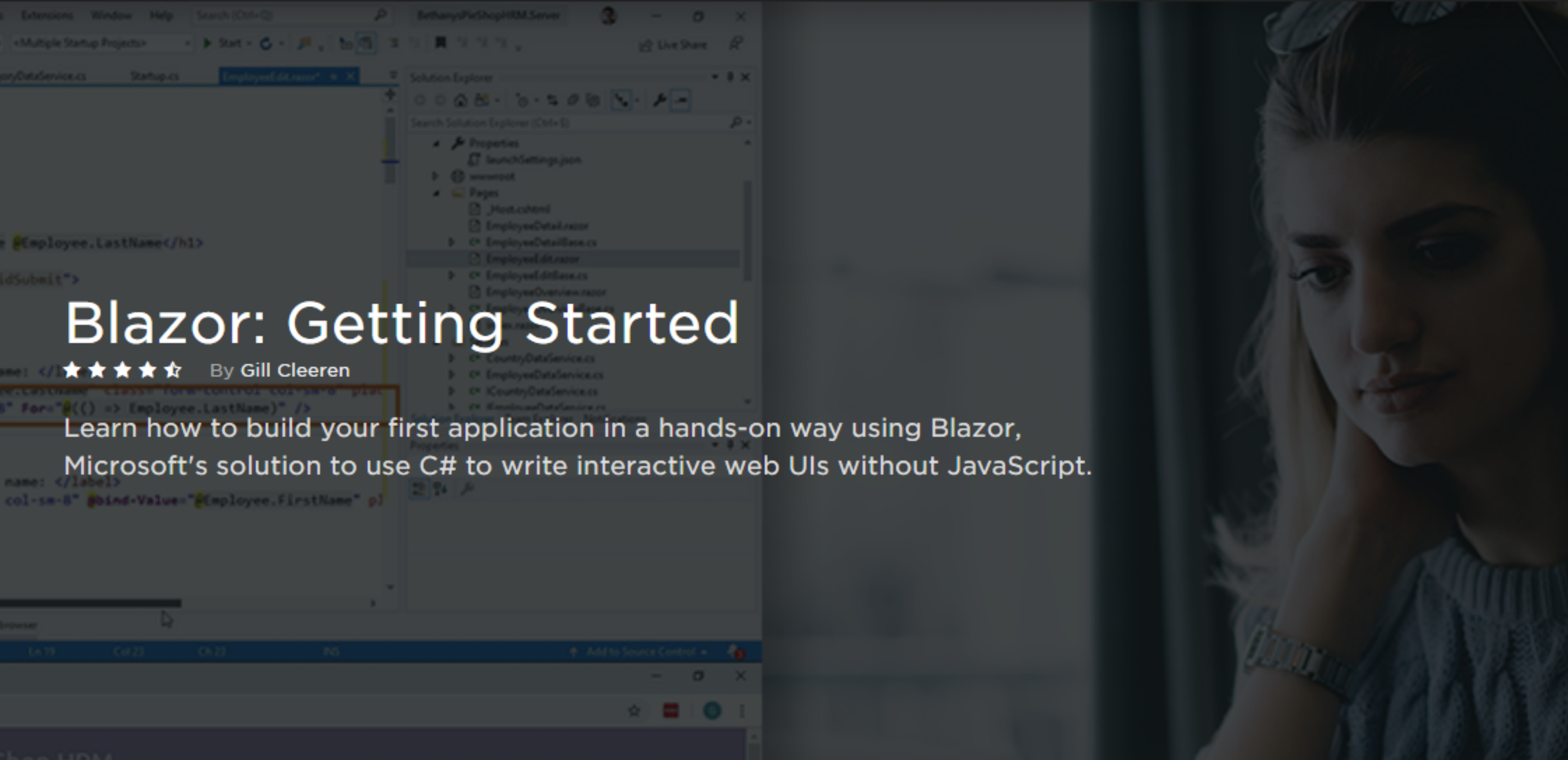
Benefits of Visual Studio and .NET including performance and libraries

Client-side Blazor



Server-side Blazor





Blazor: Getting Started

★★★★☆ By Gill Cleeren

Learn how to build your first application in a hands-on way using Blazor, Microsoft's solution to use C# to write interactive web UIs without JavaScript.

Using NSwag and NSwagStudio

Accessing the API

We can write all code manually for each type of client we'll want to connect.

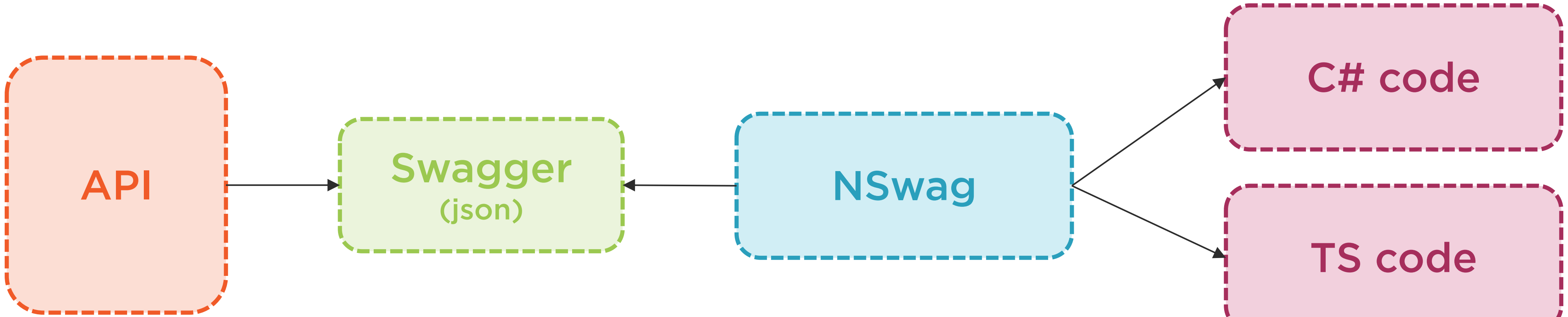
Using NSwag

Tooling based on Swagger/OpenAPI specification

Generation of client code to use the API

Works for many technologies including .NET Core and TypeScript

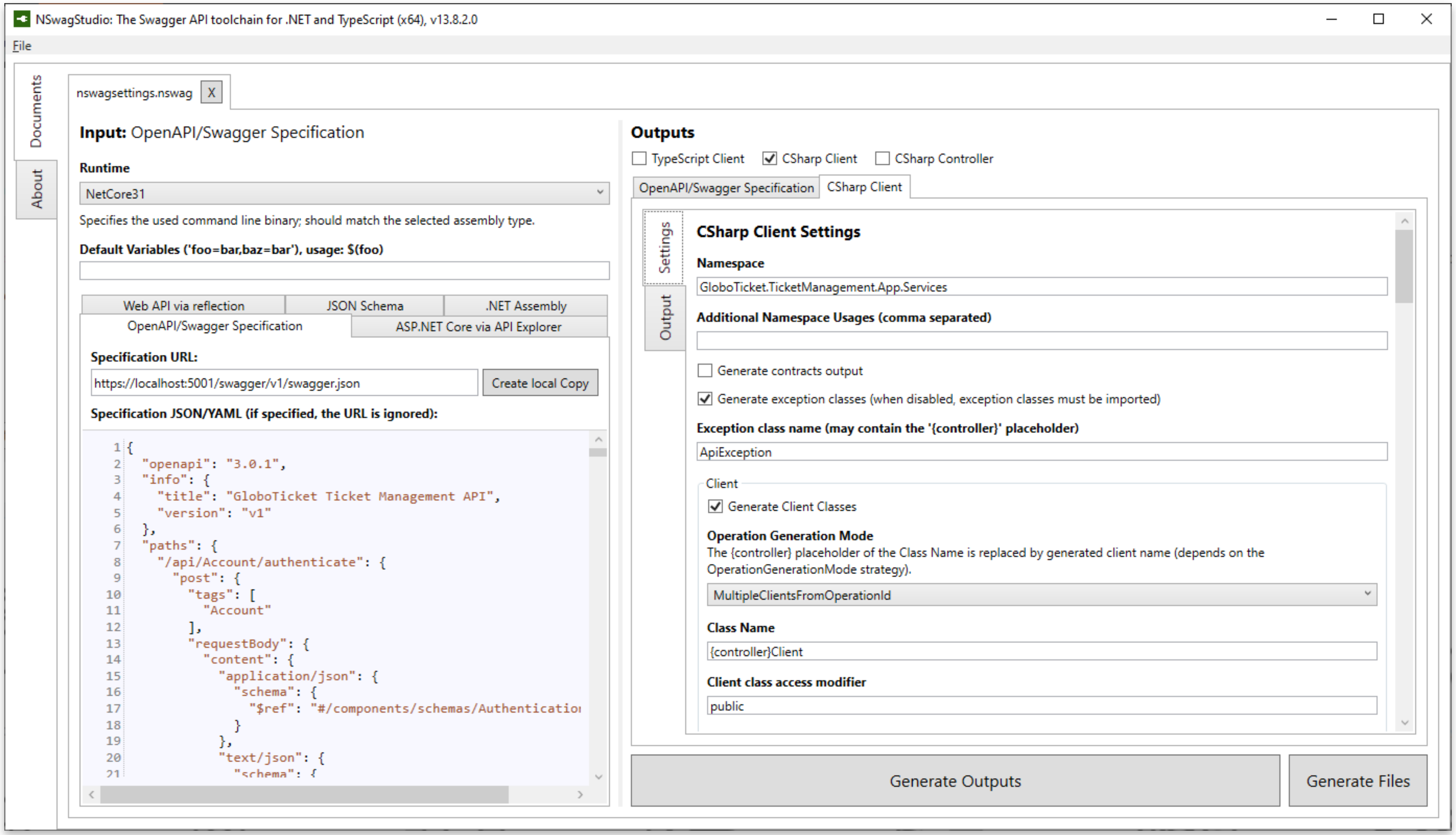
Using NSwag



Using NSwag

- NSwagStudio
- Build
- CLI

NSwagStudio


NSwagStudio: The Swagger API toolchain for .NET and TypeScript (x64), v13.8.2.0
File
Documents
About
nswagsettings.nswag X

Input: OpenAPI/Swagger Specification

Runtime

NetCore31

Specifies the used command line binary; should match the selected assembly type.

Default Variables ('foo=bar,baz=bar'), usage: $(foo)

Web API via reflection
JSON Schema
.NET Assembly
OpenAPI/Swagger Specification
ASP.NET Core via API Explorer

Specification URL:

https://localhost:5001/swagger/v1/swagger.json

Specification JSON/YAML (if specified, the URL is ignored):

```
{
  "openapi": "3.0.1",
  "info": {
    "title": "GloboTicket Ticket Management API",
    "version": "v1"
  },
  "paths": {
    "/api/Account/authenticate": {
      "post": {
        "tags": [
          "Account"
        ],
        "requestBody": {
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/Authenticatio
              }
            },
            "text/json": {
              "schema": {
```

Outputs

☐ TypeScript Client ☑ CSharp Client ☐ CSharp Controller

CSharp Client Settings

Namespace

GloboTicket.TicketManagement.App.Services

Additional Namespace Usages (comma separated)

☐ Generate contracts output

☑ Generate exception classes (when disabled, exception classes must be imported)

Exception class name (may contain the '{controller}' placeholder)

ApiException

Client

☑ Generate Client Classes

Operation Generation Mode

The {controller} placeholder of the Class Name is replaced by generated client name (depends on the OperationGenerationMode strategy).

MultipleClientsFromOperationId

Class Name

{controller}Client

Client class access modifier

public

Demo

Using NSwagStudio to generate client code

Exploring the Client App

The Blazor Client App

Packages

AutoMapper

HttpClientFactory

NSwag

Service code

Authentication

Covered in next module

Demo

Exploring the Blazor application

“The list of orders is now a bit too long. Can we get it paged?”

Demo

Adding paging functionality on the API

Generating new code with NSwag

Updating the Blazor app

Summary

NSwag works well with Swagger API

- Less code to write

Blazor can plug in on our architecture

Up next:

Handling cross-cutting concerns