

Mariusz Wieczorek

mariusz.wieczorek@kabat.pl

Streszczenie

Wdrażanie aplikacji  
Serwer IIS  
Porady

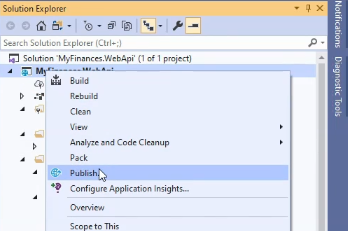
WEB API

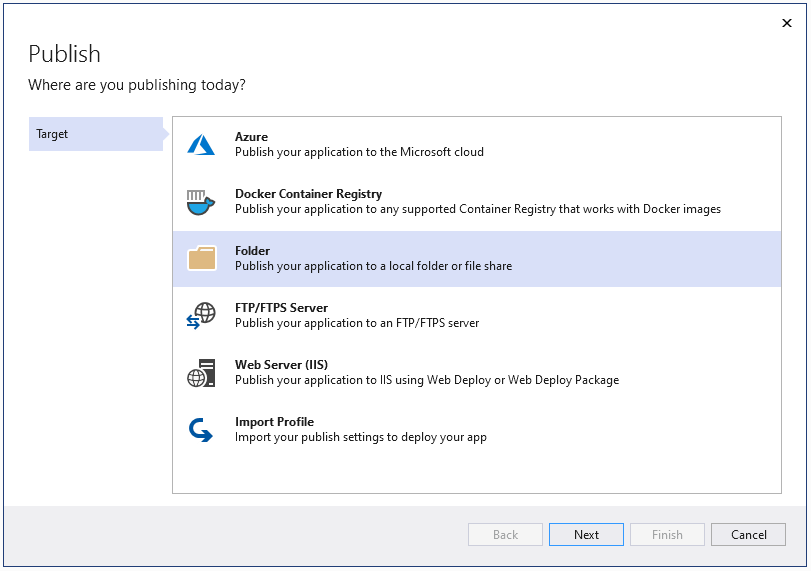
Wdrażanie Aplikacji na IIS

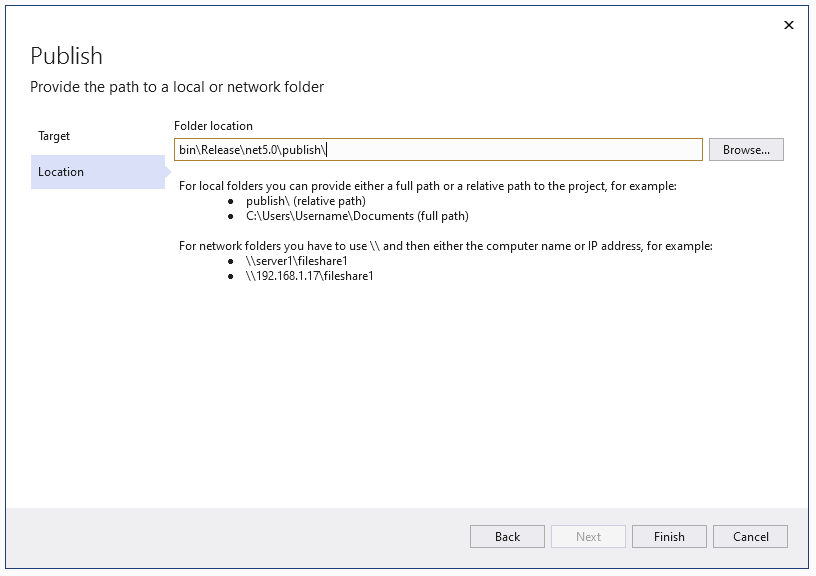
**Wdrażanie Aplikacji**

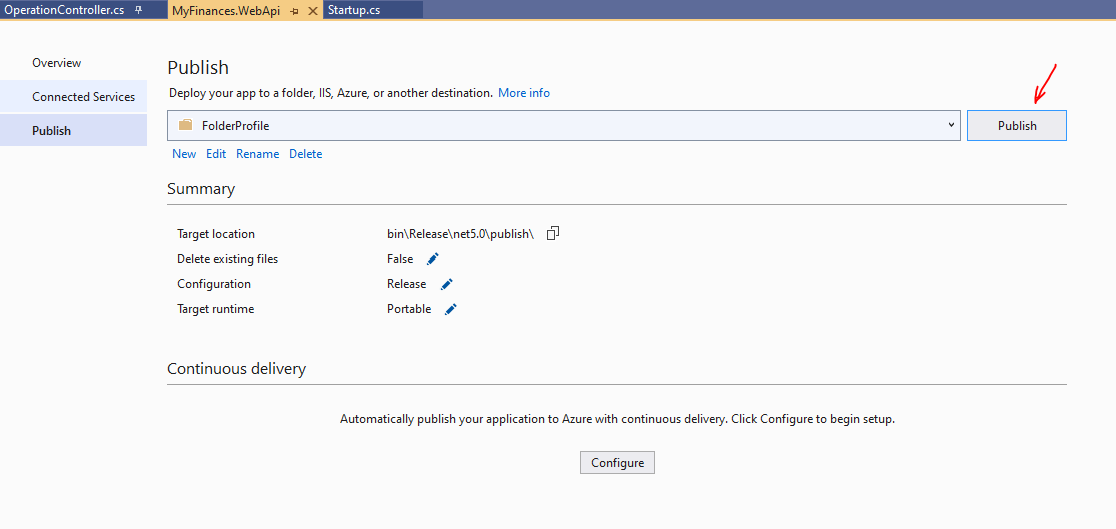


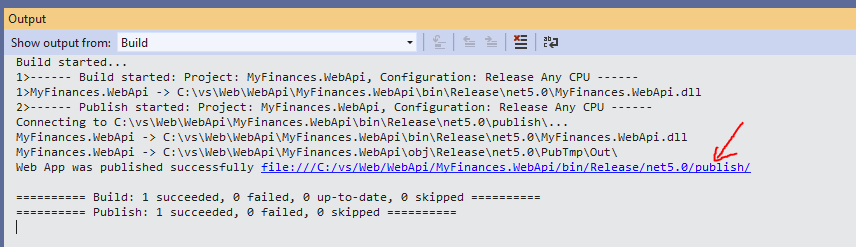
Klikamy na nazwie projektu wybieramy Publish



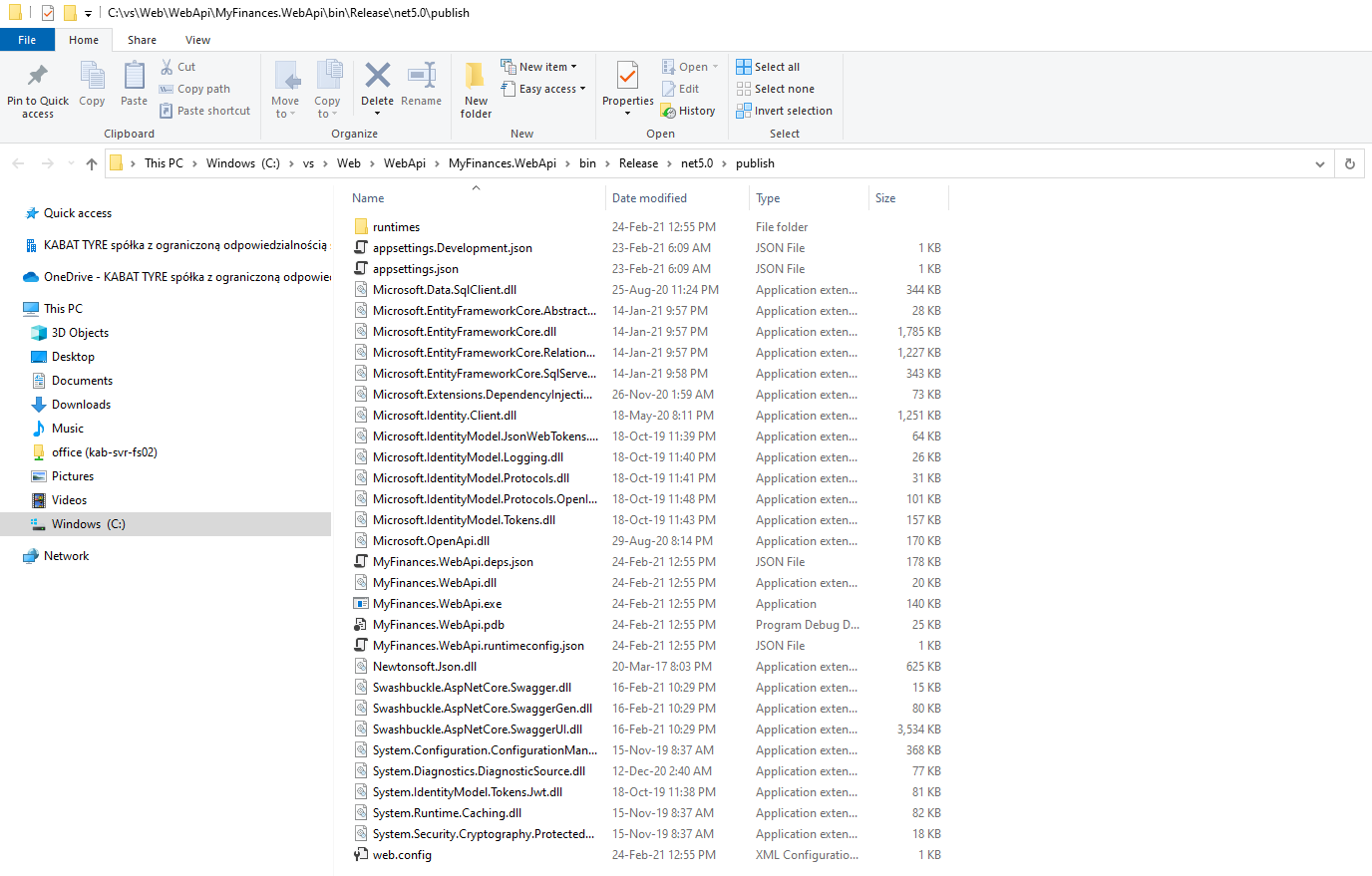




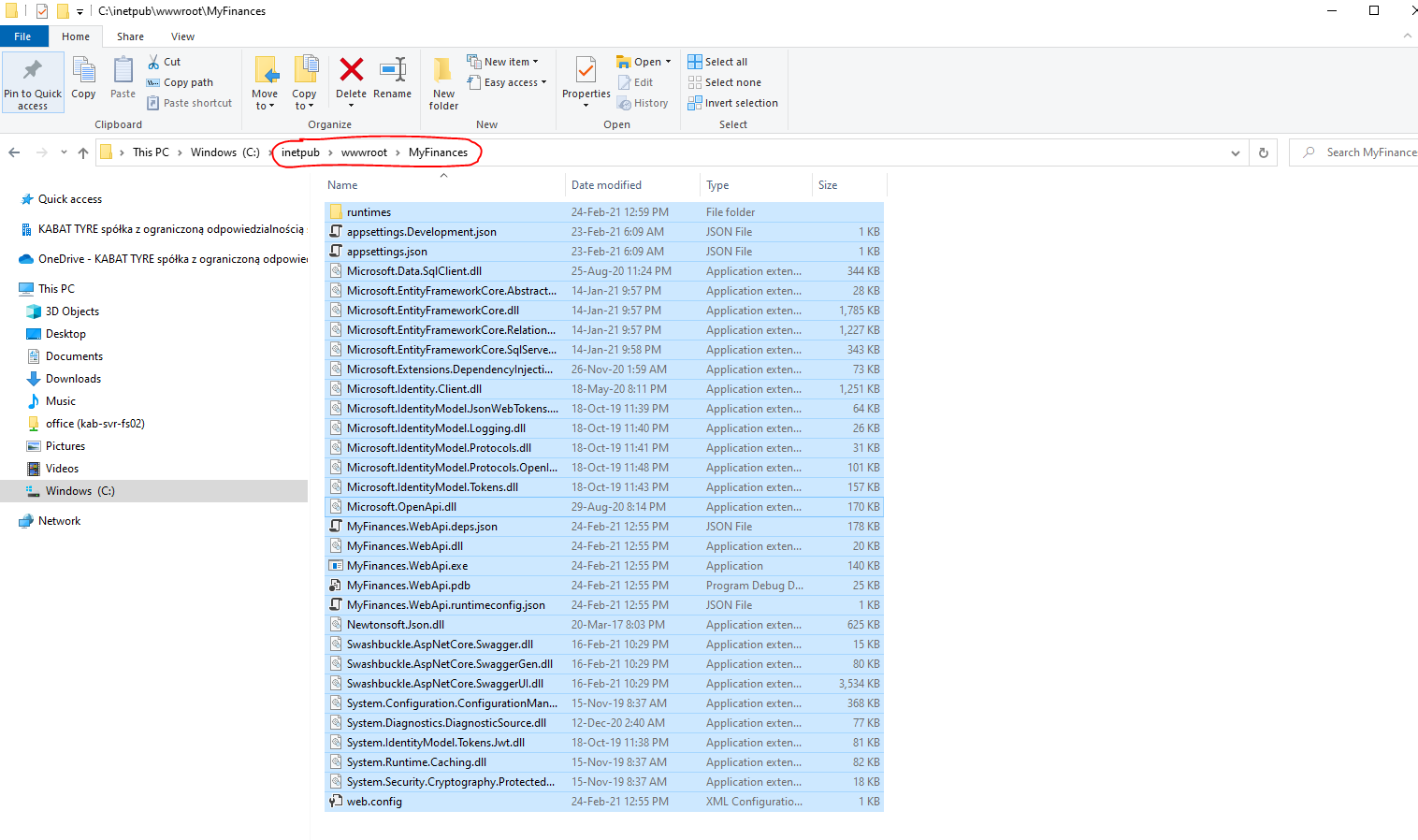


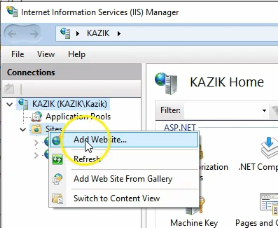


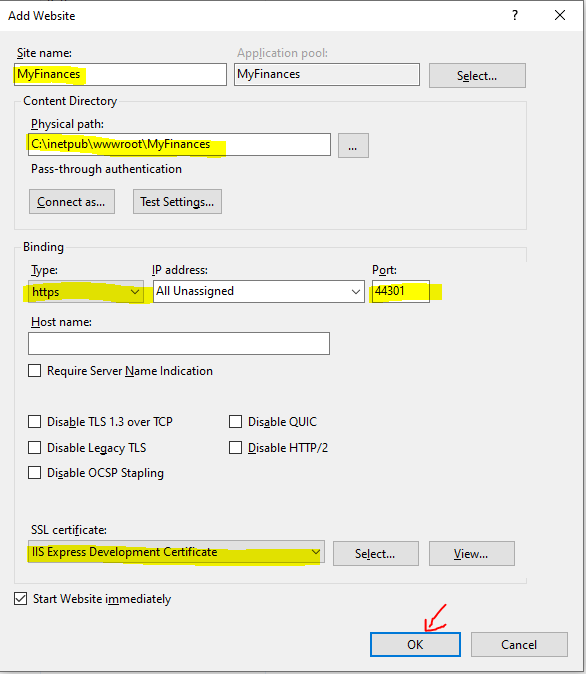
Klikamy na linku (Ctrl+LMB) aby otworzyć folder z publikacją



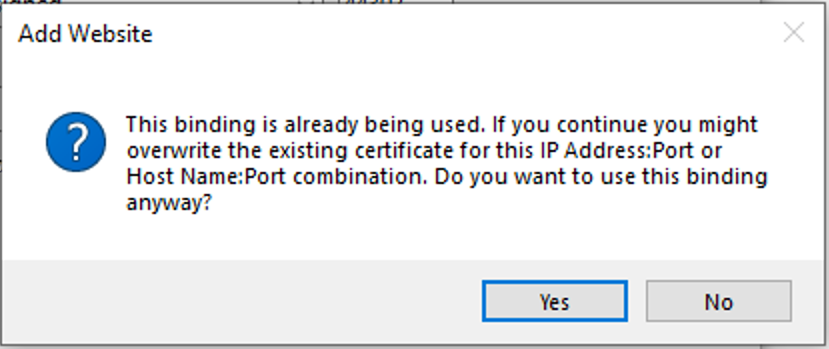
Kopiujemy pliki, do folderu w którym mamy nasze aplikacje  
W IIS klikamy na Default Web Site i wybieramy Explore, Tam tworzymy folder np. MyFinances  
i wgrywamy pliki naszej publikacji







|  |
| --- |
| **Uwaga** na ostrzeżenia dotyczące zajętych portów 44301 … |



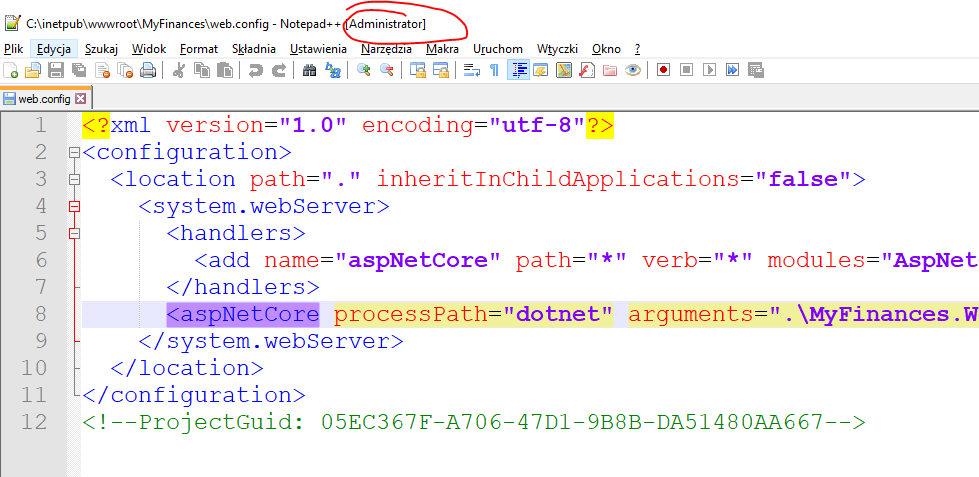
**Problemy z wdrażaniem i diagnozowanie błędów**

|  |
| --- |
| **Uwaga**  W przypadku aplikacji korzystającej z .NET 5.0 lub pewnie innych z **core** należy doinstalować **hosting bundle**. A najlepiej niezależnie od wersji zainstalować tą paczkę.  <https://dotnet.microsoft.com/download/dotnet/thank-you/runtime-aspnetcore-3.1.12-windows-hosting-bundle-installer> |

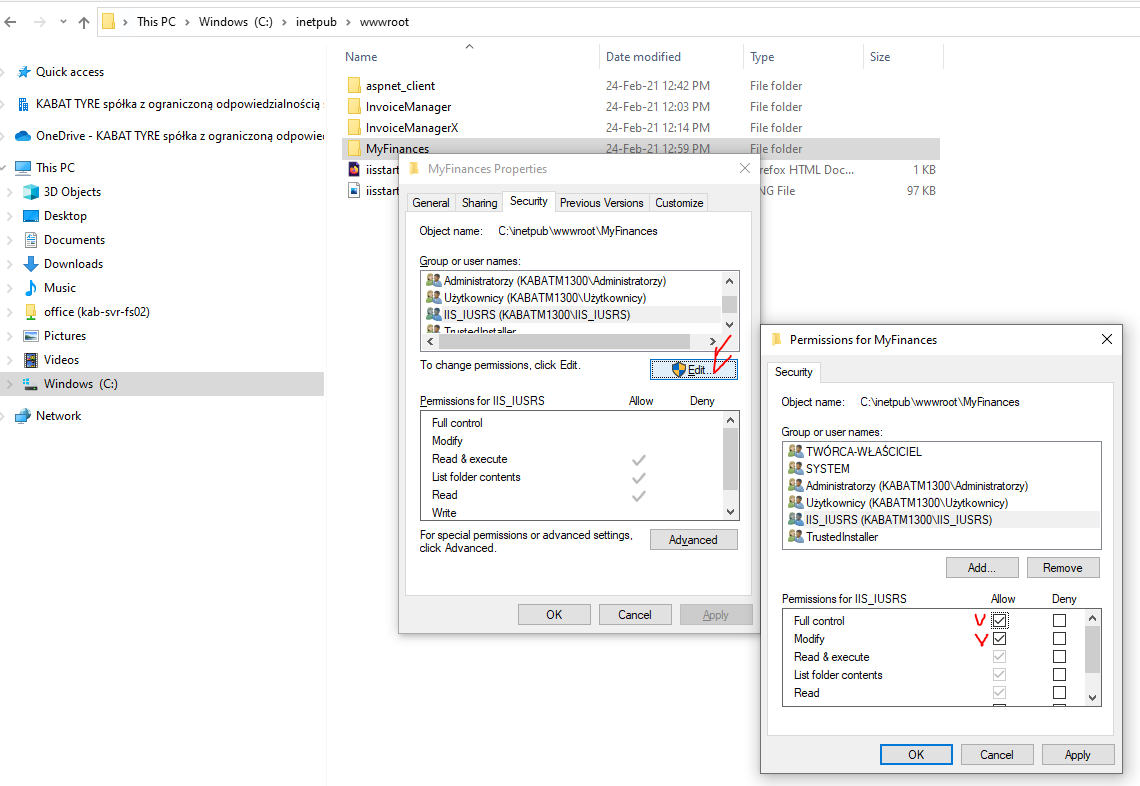
**Diagnozowanie błędów**

Otwieramy za pomocą notepad++ plik Web.config, musimy otworzyć notatnik z uprawnieniami  
administratora, ustawiamy stdoutLogEnabled="true"

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <configuration>  <location path="." inheritInChildApplications="false">  <system.webServer>  <handlers>  <add name="aspNetCore" path="\*" verb="\*" modules="AspNetCoreModuleV2" resourceType="Unspecified" />  </handlers>  <aspNetCore processPath="dotnet" arguments=".\MyFinances.WebApi.dll"   stdoutLogEnabled="true" stdoutLogFile=".\logs\stdout" hostingModel="inprocess" />  </system.webServer>  </location>  </configuration>  <!--ProjectGuid: 05EC367F-A706-47D1-9B8B-DA51480AA667--> |



Aby IIS mógł pisać do pliku z logiem musimy nadać mu uprawnienia.



**Porady**

Metod HttpGet (i innych) może być nieskończenie wiele, ważne żeby dało się je rozróżnić.  
Nazwa metody HttpGet to znaczy "GET" jest dowolna, ta metoda równie dobrze może nazywać się GetOperations itp i dalej zadziała. Ważne, żeby ta ścieżka była inna:

[HttpGet("{recordsPerPage}/{pageNr}")]

Ale to można łatwo zrobić, możesz np mieć inną metodę taką:

[HttpGet("getoperations/{recordsPerPage}/{pageNr}")]

inną taką:

[HttpGet("getoperationsX/{recordsPerPage}/{pageNr}")]

itd :)

Na podstawie ścieżki w tym atrybucie możesz się dostać do tej akcji, która Cię interesuje.

**Paginacja**

Kontroler

|  |
| --- |
| [HttpGet("{recordsPerPage}/{pageNr}")]  public DataResponse<IEnumerable<OperationDto>> Get(int recordsPerPage, int pageNr)  {  var responce = new DataResponse<IEnumerable<OperationDto>>();  try  {  responce.Data = \_unitOfWork.Operation.Get(recordsPerPage, pageNr).ToDtos();  }  catch (Exception ex)  {  // logowanie do pliku ...  responce.Errors.Add( new Error(ex.Source,ex.Message));  }    return responce;  } |

Repozytorium

|  |
| --- |
| public IEnumerable<Operation> Get(int recordsPerPage, int pageNr)  {  if (pageNr <= 0)  {  pageNr = 1;  }    if (recordsPerPage <= 0)  {  recordsPerPage = 1;  }  int recordToSkip = recordsPerPage \* (pageNr - 1);  return \_context.Operations.Skip(recordToSkip).Take(recordsPerPage);  } |