Why error handling ?

Agenda

.NET solution files

The astronomical calculation library

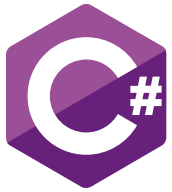
Mass conversion methods and its errors

Gravity calculation methods and its errors

Calling the library from the api

Why error handling ?

.NET solutions

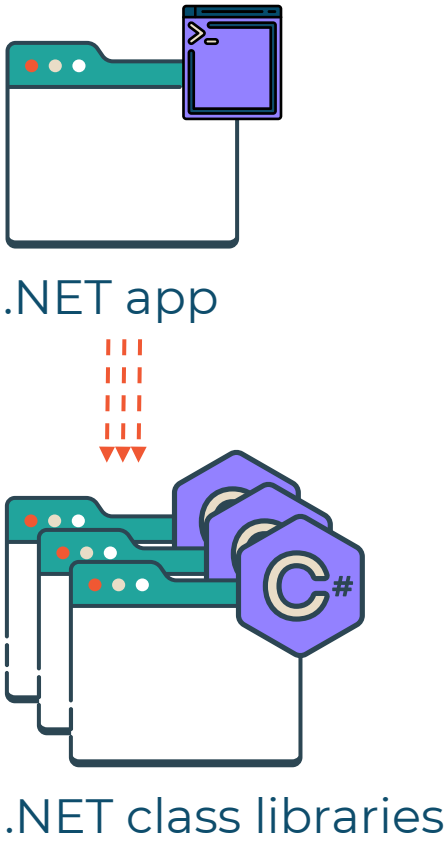


.NET projects

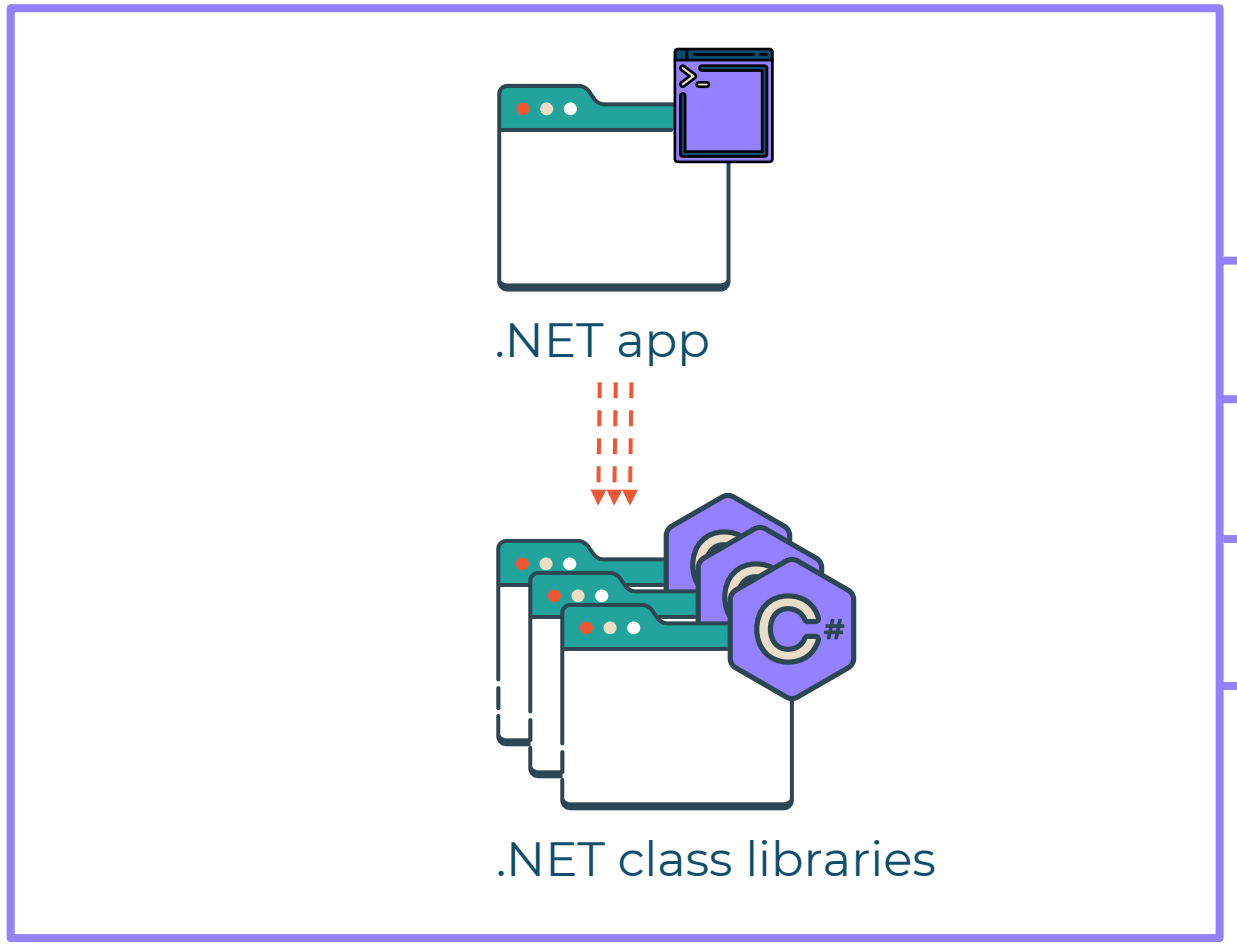
Generates an executable or a dynamic link library (assemblies)

Contains compilation units (cs files), configuration files (.config, .json) and resources (images, icons ...)

Generally, a .NET application includes multiple projects (App + referenced libraries)



Solution = project container



Demo

Create a solution using the CLI

Create a solution using an extension

Demo

Astronomical calculator challenge reminder

Demo

Call mass conversion methods from console

Run with valid input data

Run with invalid input data, show error conditions

Demo

Call the gravity calculation methods from console

Run with valid input data

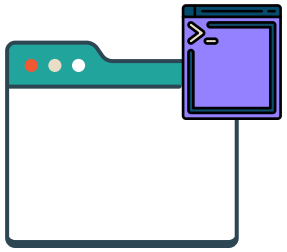
Run with invalid input data, show error conditions

Demo

Call library methods from a .NET API

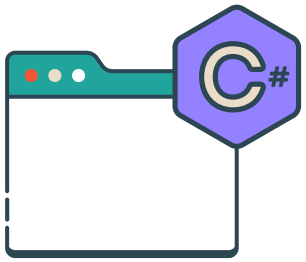
Call the API with valid input data

Run the API with invalid input data, show error conditions



dotnet run

API app



Calculation library

http://localhost:5000/

Calculate

CalculateForPlanet

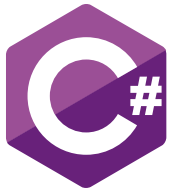
API endpoints

get http requests

http responses

Swagger client page

Why handling errors ?



Errors

Occur during execution

2 types of errors : expectable and not expectable

Error causes

Unavailable system resource

Error conditions

Time out

System failure

Memory

Error handling

Makes your code :

- More explicit
- More maintainable

Summary

A .NET solution file is a project container

Error conditions stop the execution of your program

Some errors can be predicted

You must handle errors for better code