Consultingwerk
software architecture and development

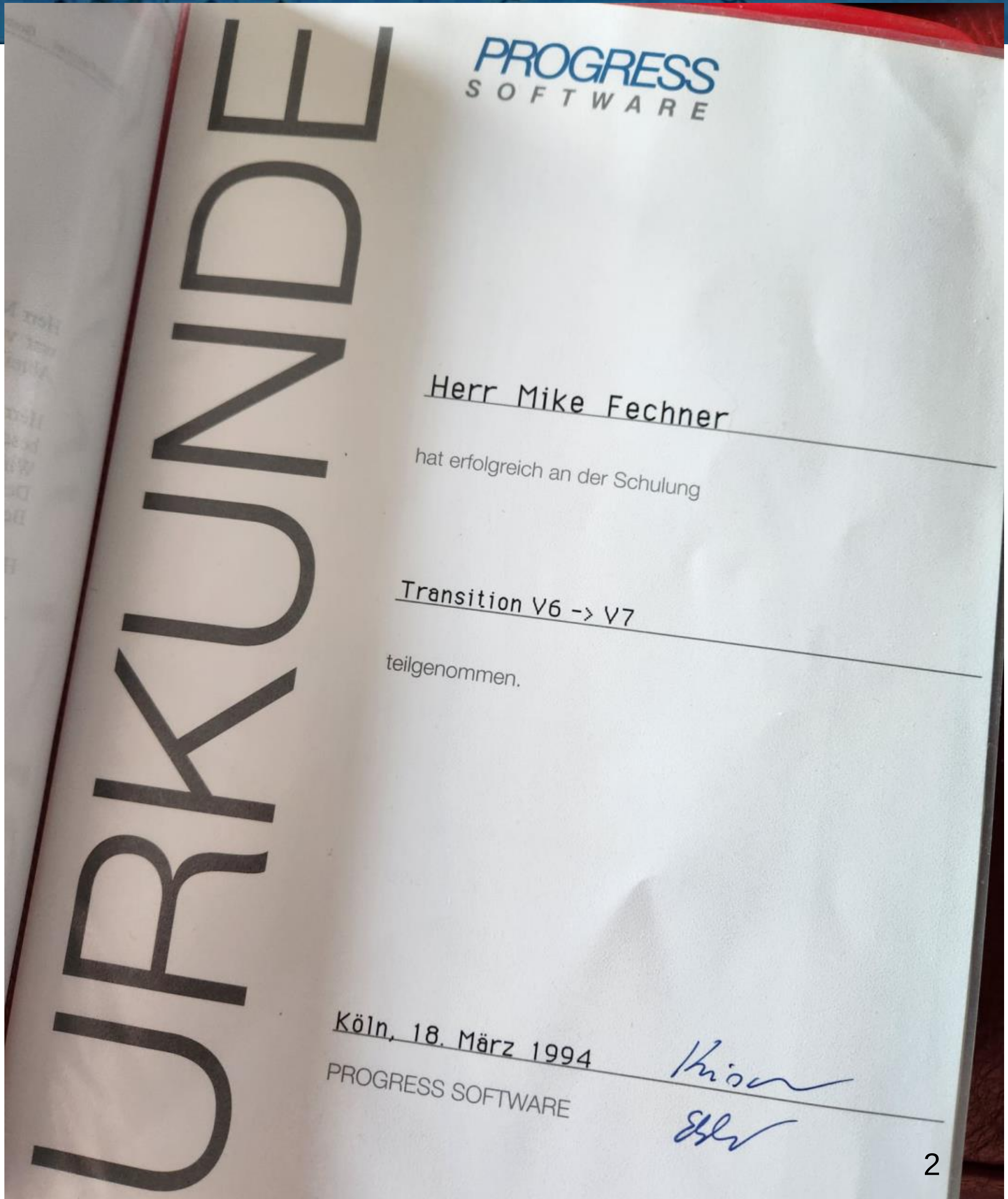
.NET Core and OpenEdge

Mike Fechner
mike.fechner@consultingwerk.de

.NET Core

Mike Fechner

- Director, Lead Modernization Architect and Product Manager, Architect of the SmartComponent Library and WinKit
- Specialized on object-oriented design, software architecture, desktop user interfaces and web technologies
- 30 years of Progress experience (V5 … OE12)
- Active member of the OpenEdge community
- Frequent speaker at OpenEdge related conferences around the world


URKUNDE

PROGRESS SOFTWARE

Herr Mike Fechner

hat erfolgreich an der Schulung

Transition V6 -> V7

teilgenommen.

Köln, 18. März 1994

PROGRESS SOFTWARE

[illegible]

[illegible]

Consultingwerk Software Services Ltd.

- Independent IT consulting organization
- Focusing on **OpenEdge** and **related technology**
- Located in Cologne, Germany, subsidiaries in UK, USA and Romania
- Customers in Europe, North America, Australia and South Africa
- Vendor of developer tools and consulting services
- Specialized in GUI for .NET, Angular, OO, Software Architecture, Application Integration
- Experts in OpenEdge Application Modernization

Services Portfolio, Progress Software

- OpenEdge (ABL, Developer Tools, Database, PASOE, …)
- Telerik DevCraft (.NET, Kendo UI, Angular, …), Telerik Reporting
- OpenEdge UltraControls (Infragistics .NET)
- Telerik Sitefinity CMS (incl. integration with OpenEdge applications)
- Kinvey Plattform, NativeScript
- Corticon BRMS
- WhatsUp Gold infrastructure-, network- and application monitoring
- Kemp Loadmaster
- …

Services Portfolio, related products

- Protop Database Monitoring
- Combit List & Label
- Web frameworks, e.g. Angular
- .NET
- Java
- ElasticSearch, Lucene
- Amazon AWS, Azure
- DevOps, Docker, Jenkins, ANT, Gradle, JIRA, …
- …

Agenda

- **.NET Core**
- .NET and OpenEdge
- Using .NET Core in OpenEdge 12.7
- Consuming asynchronous API’s
- Benchmarking

.NET Framework

.NET Framework still main stream for Windows Desktop development

- 2002 .NET Framework 1.0
- 2006 .NET Framework 2.0 (**mainstream** begins)
- 2006 .NET Framework 3.0, WPF, WCF, workflow foundation, CLR remains 2.0
- 2009 .NET Framework 4.0, new CLR
- 2019 .NET Framework 4.8
- 2022 .NET Framework 4.8.1
- .NET Framework 4.8 is the last version of .NET Framework (*but, wasn't Windows 10 the last version of Windows?*)
- Officially, .NET Framework is feature-stable

.NET / .NET Core / .NET Framwork

- Don’t shoot the messenger!

.NET Framework is not .NET

.NET is not .NET Framework

.NET Overview

.NET Framework

.NET Core

.NET Standard

.NET

.NET Core

- Announced 2014 as „Cloud-optimized .NET Framework“
- First RC was named .NET Core 5.0 to indicate continuity with .NET Framework 4.x releases
- However, it was released as .NET Core 1.0 ☺
- **Accepts breaking changes compared to .NET Framework**
- With release 5.0 “Core” was dropped from the name: .NET 5.0
- Performance improvements in .NET 6.0, further performance improvements in .NET 7.0
- Related to the Mono framework (.NET port to Linux)

.NET Core Release Cycle

https://learn.microsoft.com/de-de/lifecycle/products/microsoft-net-and-net-core

- 2016 Release .NET Core 1.0
- 2017 Release .NET Core 2.0 and Net Standard
- 2018 Release .NET Core 2.1 (LTS)
- 2019 Release .NET Core 3.0
- 2020 Release .NET Core 3.1.4 (LTS)
- 2020 .NET 5
- 2021 .NET 6 (LTS – November 2024)
- 2022 .NET 7
- 2023 .NET 8 (preview)

Annual Major-Release

Alternating LTS-Releases

.NET Core

- Large parts of the community refer to .NET 6.0 as .NET Core 6.0 – even when Microsoft is no longer using that name
- It‘s just human to distinguish .NET Core vs. .NET Framework
- Even Progress Software has named the startup parameter to enable support for .NET 6.0 in the AVM:

-clrnetcore ☺

.NET Core Details

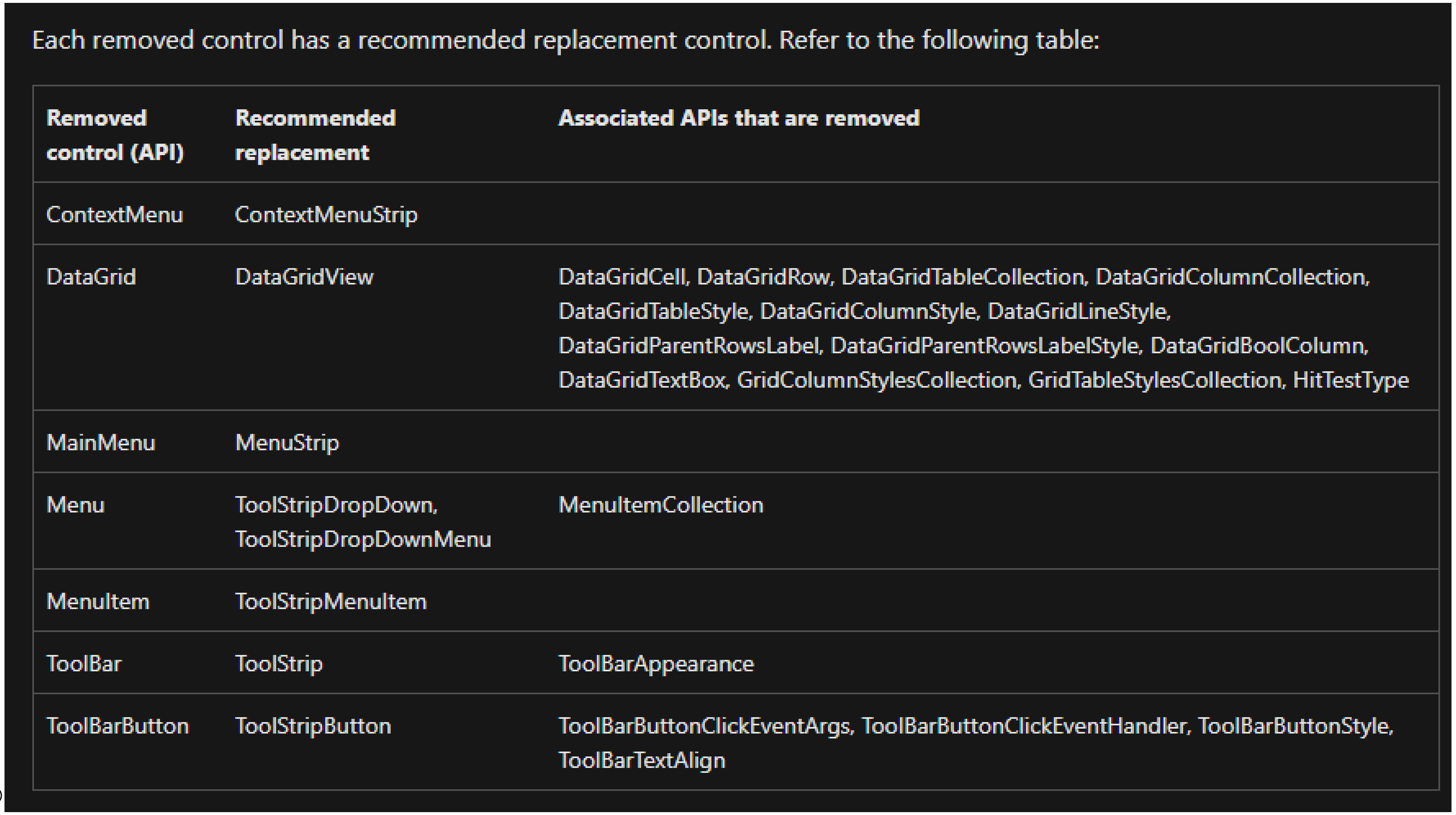
- Platform independent (Windows/Linux/macOS x86/x64, ARM)
- Open Source (MIT/Apache2) – github.com/dotnet/core
- C# / F# / VB.NET (with limitations)
- Visual Studio / Visual Studio Code / Command Line (dotnet new|build|run)
  Focus on CLI tools, micro services and Web development

.NET Core Limitations

- No Web Forms (but ASP.NET Core MVC and Blazor)
- No Workflow Foundation
- No Windows Communication Foundation
- Limited support for Windows (WPF, Windows Forms only on Windows)
  - Known breaking changes, e.g. https://learn.microsoft.com/en-us/dotnet/core/compatibility/3.1#windows-forms
  - Some Controls – especially those from .NET 1.0 that are replaced by newer variants in .NET 2.0 – no longer available
  - DataView -> DataGridView, Toolbars, Menu

Consultingwerk
software architecture and development

Removed WinForms controls in .NET Core



Each removed control has a recommended replacement control. Refer to the following table:

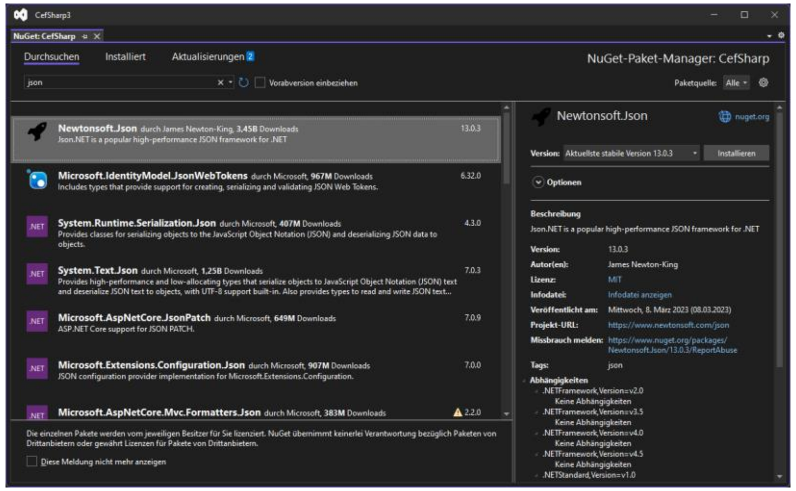
Removed control (API)	Recommended replacement	Associated APIs that are removed
ContextMenu	ContextMenuStrip	
DataGrid	DataGridView	DataGridCell, DataGridRow, DataGridTableCollection, DataGridColumnCollection, DataGridTableStyle, DataGridColumnStyle, DataGridLineStyle, DataGridParentRowsLabel, DataGridParentRowsLabelStyle, DataGridBoolColumn, DataGridTextBox, GridColumnStylesCollection, GridTableStylesCollection, HitTestType
MainMenu	MenuStrip	
Menu	ToolStripDropDown, ToolStripDropDownMenu	MenuItemCollection
MenuItem	ToolStripMenuItem	
ToolBar	ToolStrip	ToolBarAppearance
ToolBarButton	ToolStripButton	ToolBarButtonClickEventArgs, ToolBarButtonClickEventHandler, ToolBarButtonStyle, ToolBarTextAlign

.NET Core new features and benefits

- Integrated Web Server
- Simplifies containerization of applications
- .NET Framework new feature development frozen (4.8)
- Blazor (Server / WebAssembly)
- MAUI
- NuGet packet manager

NuGet package manager

- Centralized Package Management
- Gives easy way to use and update 3rd party libraries
- Manages recursive package dependencies
- Integrated into Visual Studio
- Command Line (Power Shell, msbuild, dotnet)



Newtonsoft.Json durch James Newton-King, 3,45B Downloads 13.0.3
Json.NET is a popular high-performance JSON framework for .NET

Microsoft.IdentityModel.JsonWebTokens durch Microsoft, 967M Downloads 6.32.0
Includes types that provide support for creating, serializing and validating JSON Web Tokens.

System.Runtime.Serialization.Json durch Microsoft, 407M Downloads 4.3.0
Provides classes for serializing objects to the JavaScript Object Notation (JSON) and deserializing JSON data to objects.

System.Text.Json durch Microsoft, 1,25B Downloads 7.0.3
Provides high-performance and low-allocating types that serialize objects to JavaScript Object Notation (JSON) text and deserialize JSON text to objects, with UTF-8 support built-in. Also provides types to read and write JSON text...

Microsoft.AspNetCore.JsonPatch durch Microsoft, 649M Downloads 7.0.9
ASP.NET Core support for JSON PATCH.

Microsoft.Extensions.Configuration.Json durch Microsoft, 907M Downloads 7.0.0
JSON configuration provider implementation for Microsoft.Extensions.Configuration.

Microsoft.AspNetCore.Mvc.Formatters.Json durch Microsoft, 383M Downloads 2.2.0

Die einzelnen Pakete werden vom jeweiligen Besitzer für Sie lizenziert. NuGet übernimmt keinerlei Verantwortung bezüglich Paketen von Drittanbietern oder gewährt Lizenzen für Pakete von Drittanbietern.

Diese Meldung nicht mehr anzeigen

Newtonsoft.Json

Version: Aktuellste stabile Version 13.0.3

Beschreibung

Json.NET is a popular high-performance JSON framework for .NET

Version:	13.0.3
Autor(en):	James Newton-King
Lizenz:	MIT
Infodatei:	Infodatei anzeigen
Veröffentlicht am:	Mittwoch, 8. März 2023 (08.03.2023)
Projekt-URL:	https://www.newtonsoft.com/json
Missbrauch melden:	https://www.nuget.org/packages/Newtonsoft.Json/13.0.3/ReportAbuse
Tags:	json

Abhängigkeiten
- .NETFramework,Version=v2.0
  - Keine Abhängigkeiten
- .NETFramework,Version=v3.5
  - Keine Abhängigkeiten
- .NETFramework,Version=v4.0
  - Keine Abhängigkeiten
- .NETFramework,Version=v4.5
  - Keine Abhängigkeiten
- .NETStandard,Version=v1.0

Agenda

- .NET Core
- **.NET and OpenEdge**
- Using .NET Core in OpenEdge 12.7
- Consuming asynchronous API’s
- Benchmarking

OpenEdge and .NET

- 2003 / OpenEdge 10.0: Introduction of OpenClient for .NET – wow that was quick!
- 2007 or 2008 / OpenEdge 10.2A: Introduction of OpenEdge GUI for .NET – ability to use .NET WinForms UI from OpenEdge GUI client, CLR 2.0
- 2009 / OpenEdge 10.2B: GUI bridge opened to non-GUI use-cases
- 2011 / OpenEdge 11: CLR 4.0 for GUI for .NET (CLR 2.0 cannot be used anymore)
- 2023 / OpenEdge 12.7: **Optional** use of .NET 6 runtime, .NET 4.8 default runtime

OpenEdge and .NET UI Components

- 2007 : OpenEdge UltraControls, cooperation with Infragistics, Infragistics Controls used by OpenEdge in QA, documentation, tech support
- 2014 : Progress Software acquired Telerik, primarily for web UI technology (Kendo UI, Sitefinity)
- Telerik RadControls support started with some issues, fixed relatively slowly
- For the presenter: OpenEdge UltraControls still first choice for OpenEdge GUI

Components of OpenEdge GUI for .NET

- .NET CLR loaded into OpenEdge client process (AVM and CLR side by side)
- In process communication via the “bridge”
- ABL can create instances of .NET types, set properties, invoke methods and subscribe to events
- ABL classes can inherit from .NET types, override methods and implement .NET interfaces
- Data Binding between OpenEdge Queries, Temp-Tables and ProDatasets and .NET Controls using standard .NET Data Binding

Components of OpenEdge GUI for .NET

- .NET exceptions integrated into ABL error handling
- .NET types extend the ABL:

```
DEFINE VARIABLE oForm AS System.Windows.Forms.Form NO-UNDO .
```

- ABL GUI and .NET WinForms UI coexist in the same prowin runtime
- ABL GUI can be embedded into .NET WinForms to allow user interface facelifting

OpenEdge GUI for .NET Tooling Support

- Visual Designer in Progress Developer Studio
- Code-completion in Progress Developer Studio (out of the box, or with OEDT)
- Code-completion in VS Code through Riverside‘s Language Server
- Class Browser in Progress Developer Studio
- Add Assemblies to assemblies.xml

Agenda

- .NET Core
- .NET and OpenEdge
- **Using .NET Core in OpenEdge 12.7**
- Consuming asynchronous API’s
- Benchmarking

Using .NET Core in OpenEdge

- Starting OpenEdge 12.7 OpenEdge supports .NET 6.0 alternatively to the .NET Framework
- Supported in OpenEdge GUI client (prowin, prowin32), Web Client, Character Client and PASOE
- By default, .NET Framework 4.8 is used
- Use **-clrnetcore** startup parameter to use .NET 6.0 **instead**!
- No parallel use of .NET Framework and .NET Core

OpenEdge Documentation

- What’s new in OpenEdge 12.7: https://docs.progress.com/bundle/openedge-whats-new/page/Whats-New-in-OpenEdge-12.7.html#Support-for-.NET-6-on-Windows-in-OpenEdge

- OpenEdge GUI for .NET in ABL: https://docs.progress.com/bundle/openedge-gui-for-dotnet-in-abl/page/Support-for-.NET-in-OpenEdge.html

Prerequisites

- Install .NET 6.0 Runtime or SDK
- https://dotnet.microsoft.com/en-us/
- https://winstall.app/apps/Microsoft.DotNet.SDK.6
- https://winstall.app/apps/Microsoft.DotNet.DesktopRuntime.6

- winget install --id=Microsoft.DotNet.SDK.6 -e
- winget install --id=Microsoft.DotNet.DesktopRuntime.6 -e

.NET related startup parameters

- -clrnetcore Use .NET Core instead of .NET Framework
- -preloadCLR Load CLR into OpenEdge client at startup
- -assemblies Folder that contains assemblies.xml (compile time) and assembly files (compile time and runtime)

.NET Assemblies

- .NET Core no longer uses the GAC (Global Assembly Cache) – so all Assemblies are to be provided by the Application in the –assemblies folder
- Assembly version does not longer need to be specified in assemblies.xml – if it is specified, OpenEdge will ignore it
- No support for PrivateProbingPath in the same way as with .NET Framework
    - PrivateProbingPath allows to organize .NET Framework Assemblies in sub-directories of -assemblies

.NET Assemblies

- Currently manual composition of files in –assemblies folder required
- Use nuget (or other method) to obtain assembly files
- Use Visual Studio project to compose Assemblies folder
- Copy assemblies and their dependencies into –assemblies folder

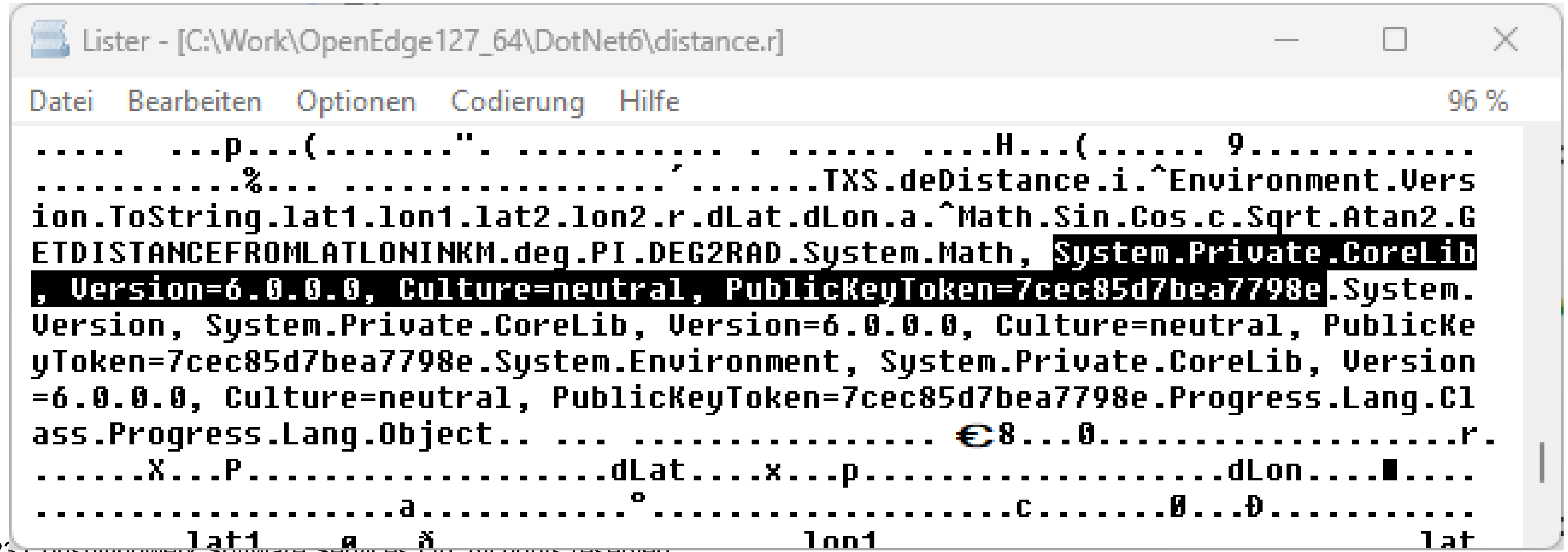
- Vote for OpenEdge idea: OPENEDGE-I-1474
- https://openedge.ideas.aha.io/ideas/OPENEDGE-I-1474

Recompile required

- ABL source code needs to be recompiled when switching from .NET Framework to .NET Core
- R-Code includes references to .NET assemblies and assemblies are not sharable between .NET Framework and .NET Core
- Recompile is required to reference correct assemblies for target .NET runtime

Assemblies referenced in R-Code

- System.IO.FileNotFoundException: The file or assembly “System.Private.CoreLib, Version 6.0.0.0” or one of its dependencies cannot be found.



Lister - [C:\Work\OpenEdge127_64\DotNet6\distance.r]

```
.....  ...p...(.......". ........... . ...... ....H...(...... 9...........
...........%... ..................´.......TXS.deDistance.i.^Environment.Vers
ion.ToString.lat1.lon1.lat2.lon2.r.dLat.dLon.a.^Math.Sin.Cos.c.Sqrt.Atan2.G
ETDISTANCEFROMLATLONINKM.deg.PI.DEG2RAD.System.Math, System.Private.CoreLib
, Version=6.0.0.0, Culture=neutral, PublicKeyToken=7cec85d7bea7798e.System.
Version, System.Private.CoreLib, Version=6.0.0.0, Culture=neutral, PublicKe
yToken=7cec85d7bea7798e.System.Environment, System.Private.CoreLib, Version
=6.0.0.0, Culture=neutral, PublicKeyToken=7cec85d7bea7798e.Progress.Lang.Cl
ass.Progress.Lang.Object.. ... ................ €8...0..................r.
......X...P...................dLat....x...p.................dLon....█....
......................a...........°.....................c.......0...Ð..........
        lat1     ø    ð                 lon1                             lat
```

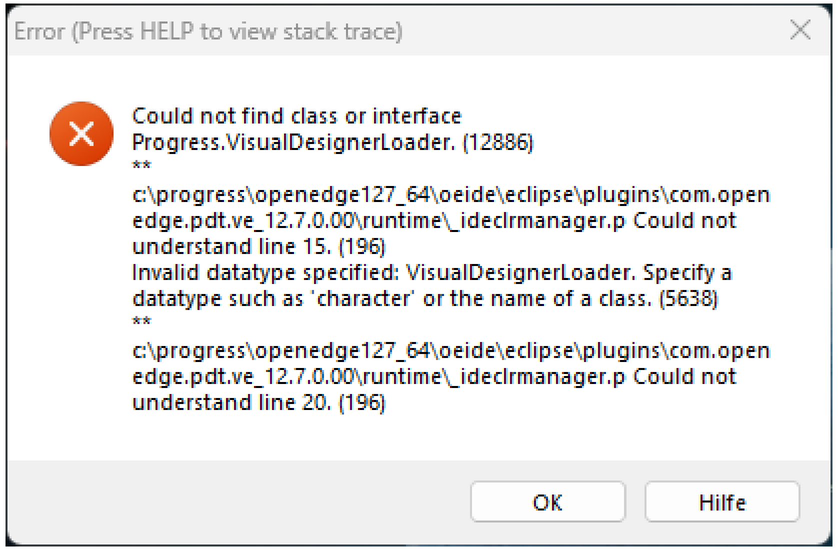
No tooling support for .NET Core in OpenEdge

- No GUI for .NET Visual Designer
- No code-completion in Progress Developer Studio
- No class-browser support for .NET Core
- No assemblies.xml maintenance
- Riverside has implemented.NET Core support in the VS Code language server for OpenEdge

- However, … the API differences between .NET Framework 4.8 and .NET 6.0 are not that significant.

Lack of GUI for .NET Visual Designer

- That‘s a bummer.



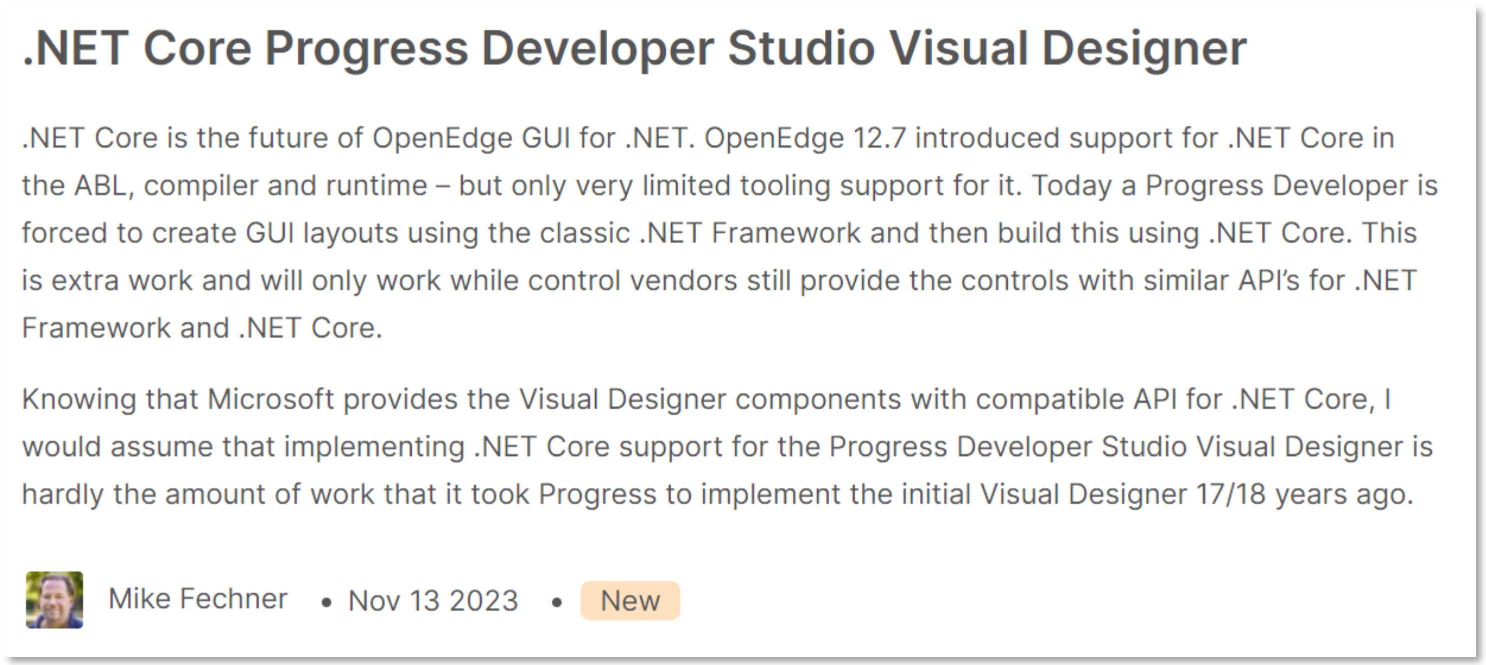
Error (Press HELP to view stack trace)

```
Could not find class or interface
Progress.VisualDesignerLoader. (12886)
**
c:\progress\openedge127_64\oeide\eclipse\plugins\com.open
edge.pdt.ve_12.7.0.00\runtime\_ideclrmanager.p Could not
understand line 15. (196)
Invalid datatype specified: VisualDesignerLoader. Specify a
datatype such as 'character' or the name of a class. (5638)
**
c:\progress\openedge127_64\oeide\eclipse\plugins\com.open
edge.pdt.ve_12.7.0.00\runtime\_ideclrmanager.p Could not
understand line 20. (196)
```

- Microsoft has implemented a new Visual Designer for .NET Core
- Updated bootstrap process will probably require Progress do so some changes to the Visual Designer in Progress Developer Studio

Please support my OpenEdge idea

- https://openedge.ideas.aha.io/ideas/OPENEDGE-I-1547



.NET Core Progress Developer Studio Visual Designer

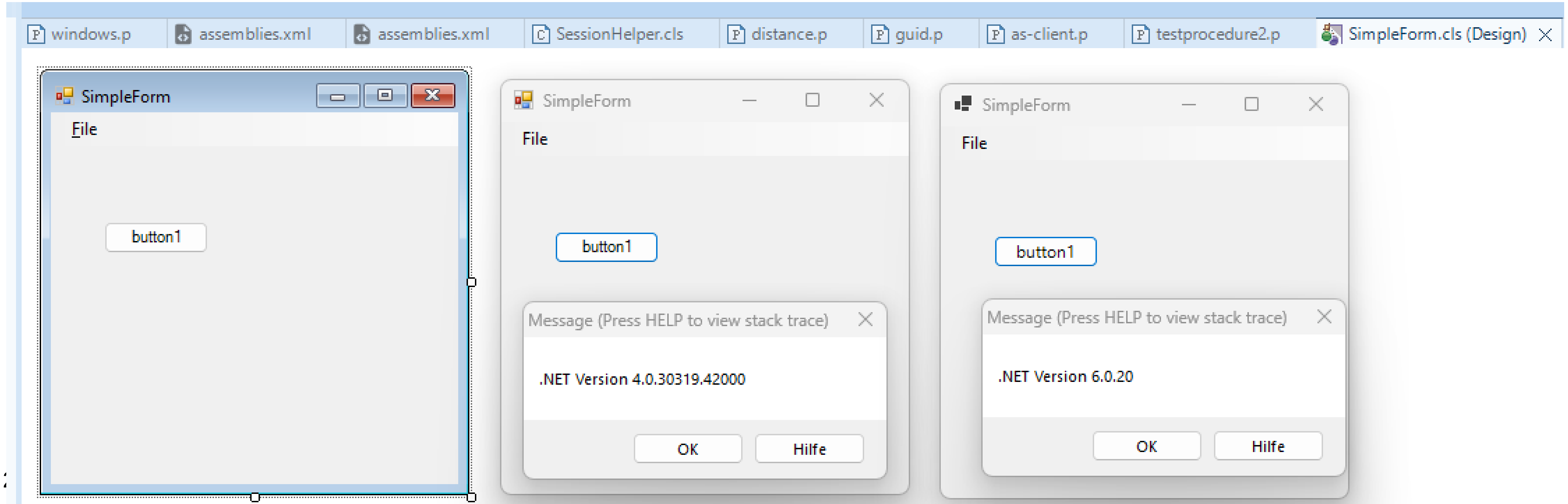
.NET Core is the future of OpenEdge GUI for .NET. OpenEdge 12.7 introduced support for .NET Core in the ABL, compiler and runtime – but only very limited tooling support for it. Today a Progress Developer is forced to create GUI layouts using the classic .NET Framework and then build this using .NET Core. This is extra work and will only work while control vendors still provide the controls with similar API's for .NET Framework and .NET Core.

Knowing that Microsoft provides the Visual Designer components with compatible API for .NET Core, I would assume that implementing .NET Core support for the Progress Developer Studio Visual Designer is hardly the amount of work that it took Progress to implement the initial Visual Designer 17/18 years ago.

Mike Fechner • Nov 13 2023 • New

Lack of GUI for .NET Visual Designer

- It is possible to use the .NET Framework Designer to design for .NET Core
- Source Code typically compatible



Message (Press HELP to view stack trace)

.NET Version 4.0.30319.42000

Message (Press HELP to view stack trace)

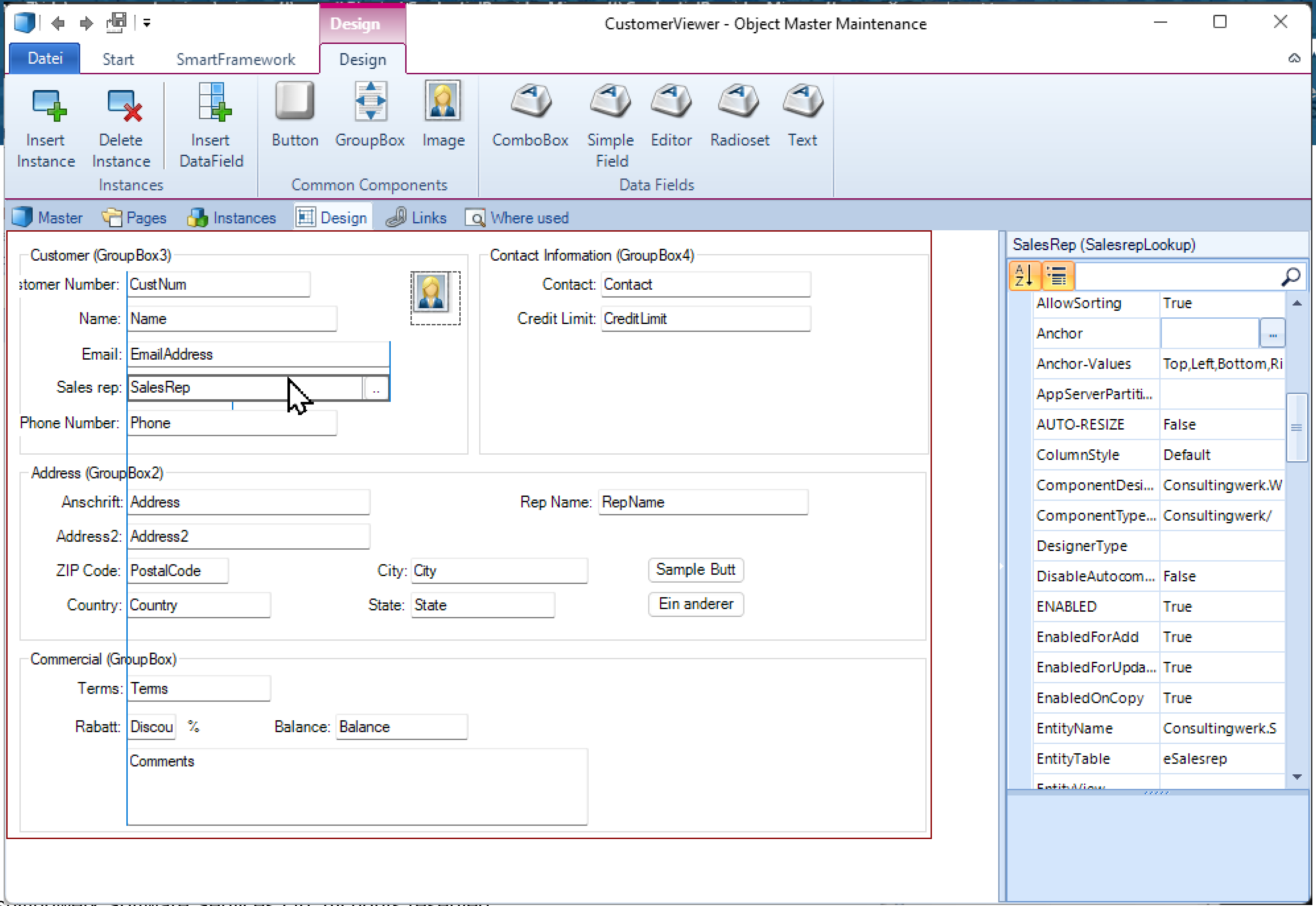
.NET Version 6.0.20

Demo

- Using .NET Framework Progress Developer Studio Visual Designer and executing with .NET Core

Lack of GUI for .NET Visual Designer

- In the SmartComponent Library we have implemented our own Visual Designer
- Application developers typically use our designer for standard screens and Progress Developer Studio Visual Designer only for specialized user controls
- Screen design stored in a repository database
- Same UI designer tooling for GUI and Angular web applications


CustomerViewer - Object Master Maintenance

Datei | Start | SmartFramework | Design

Insert Instance | Delete Instance | Insert DataField — Instances
Button | GroupBox | Image — Common Components
ComboBox | Simple Field | Editor | Radioset | Text — Data Fields

Master | Pages | Instances | Design | Links | Where used

Customer (GroupBox3)
stomer Number: CustNum
Name: Name
Email: EmailAddress
Sales rep: SalesRep
Phone Number: Phone

Contact Information (GroupBox4)
Contact: Contact
Credit Limit: CreditLimit

Address (GroupBox2)
Anschrift: Address
Address2: Address2
ZIP Code: PostalCode
Country: Country
City: City
State: State
Rep Name: RepName
Sample Butt
Ein anderer

Commercial (GroupBox)
Terms: Terms
Rabatt: Discou %
Balance: Balance
Comments

SalesRep (SalesrepLookup)

AllowSorting	True
Anchor	
Anchor-Values	Top,Left,Bottom,Ri
AppServerPartiti...	
AUTO-RESIZE	False
ColumnStyle	Default
ComponentDesi...	Consultingwerk.W
ComponentType...	Consultingwerk/
DesignerType	
DisableAutocom...	False
ENABLED	True
EnabledForAdd	True
EnabledForUpda...	True
EnabledOnCopy	True
EntityName	Consultingwerk.S
EntityTable	eSalesrep

SmartComponent Library Visual Designer

- Phase 1 for us: Using .NET Framework Visual Designer and .NET Core at runtime
- Phase 2 for us: Porting Visual Designer to .NET Core

OpenEdge UltraControls

- OpenEdge does not ship with OpenEdge UltraControls for .NET Core
- Components need to be obtained from Infragistics directly
- Typically installed with nuget
- Infragistics is using a different naming pattern for Assemblies, v22.1 suffix in Assembly name no longer used with .NET Core versions

.NET Libraries typically used by Consultingwerk

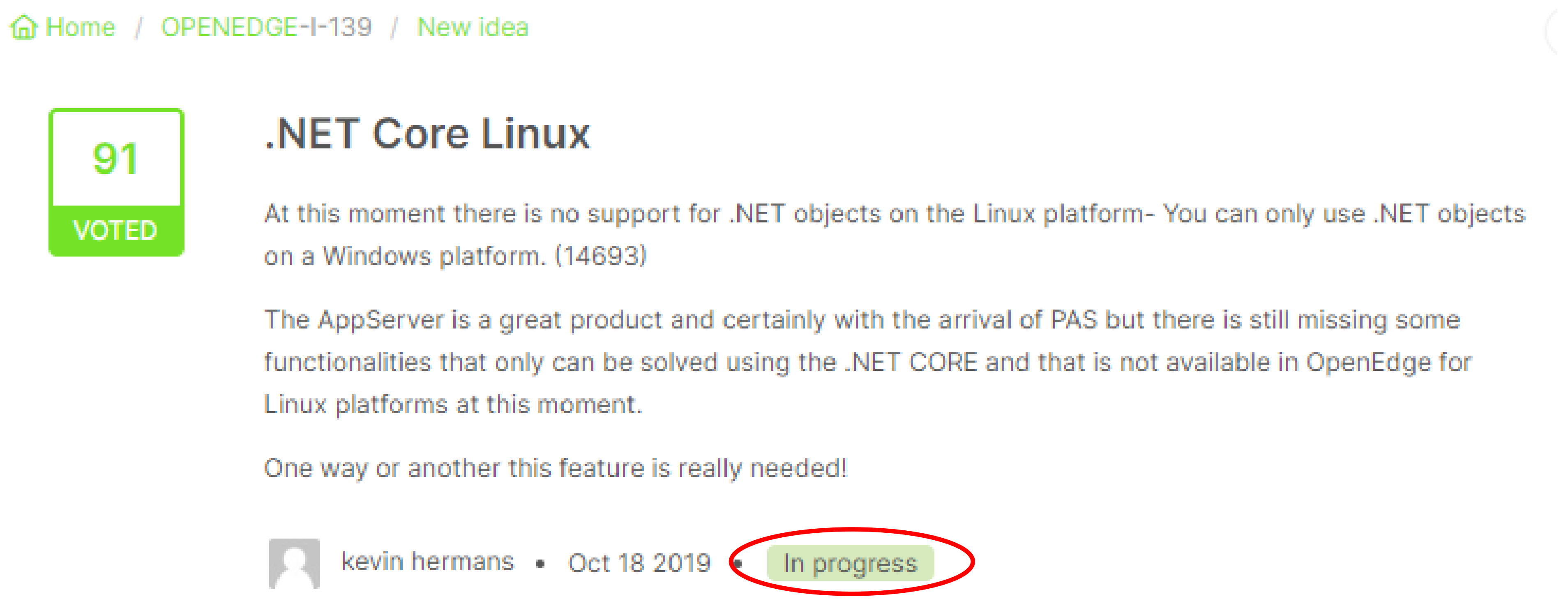
- OpenEdge UltraControls / Infragistics – available as .NET Core
- Telerik Rad Controls – available as .NET Core
- **Proparse (IKVM JVM in .NET) – recently revitalized with .NET Core support**
- CefSharp – available as .NET Core
- Various Microsoft Assemblies – available as .NET Core
- Crainiate Diagramming (ERD Designer Component) – planning to port this ourselves
- Looks like there is no real hard blocker for us

.NET Core for OpenEdge – why?

- Latest version of .NET
- Future-proofing application development
- It is expected that in the foreseeable future modern libraries will be increasingly available for .NET Core – and no longer the “legacy” .NET Framework
- Faster …
- And …

.NET Core support on Linux

- https://openedge.ideas.aha.io/ideas/OPENEDGE-I-139


Home / OPENEDGE-I-139 / New idea

91
VOTED

.NET Core Linux

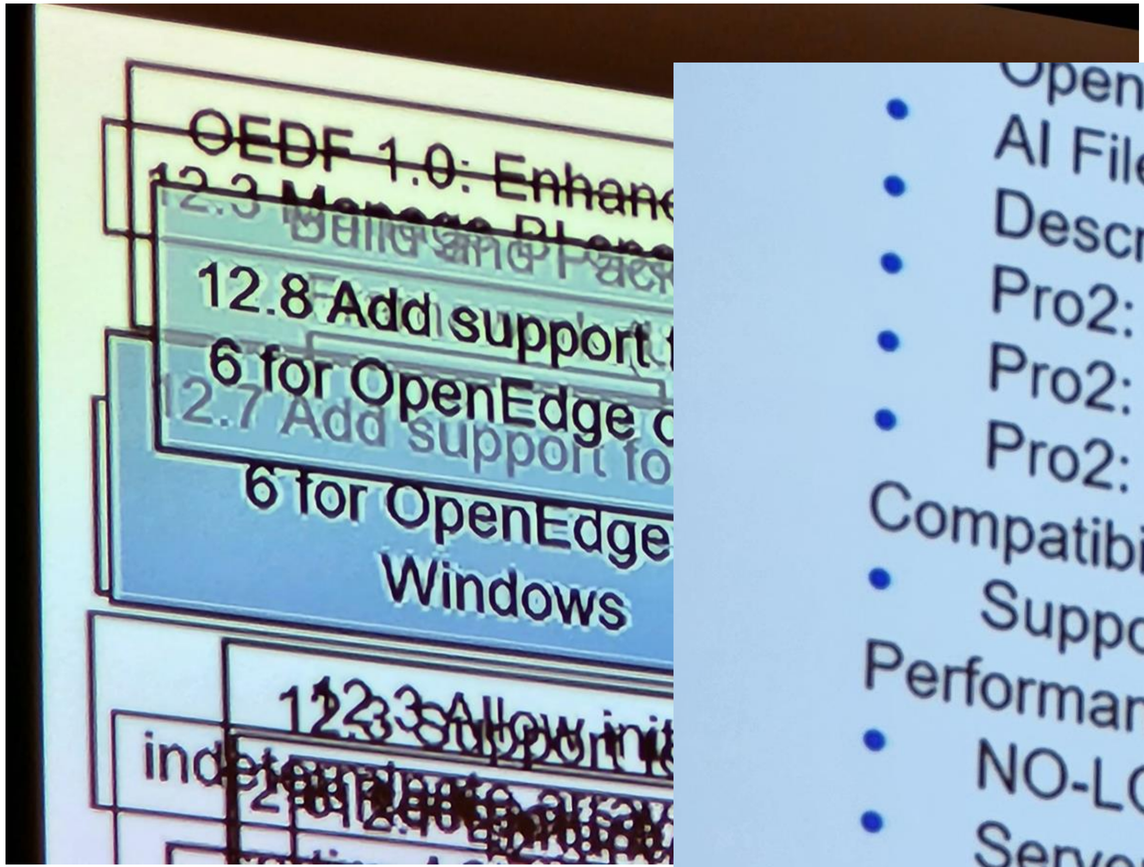
At this moment there is no support for .NET objects on the Linux platform- You can only use .NET objects on a Windows platform. (14693)

The AppServer is a great product and certainly with the arrival of PAS but there is still missing some functionalities that only can be solved using the .NET CORE and that is not available in OpenEdge for Linux platforms at this moment.

One way or another this feature is really needed!

kevin hermans • Oct 18 2019 • In progress

Attended the OpenEdge 12.8 session ???



OEDF 1.0: Enhanc

12.3 Manage Plac

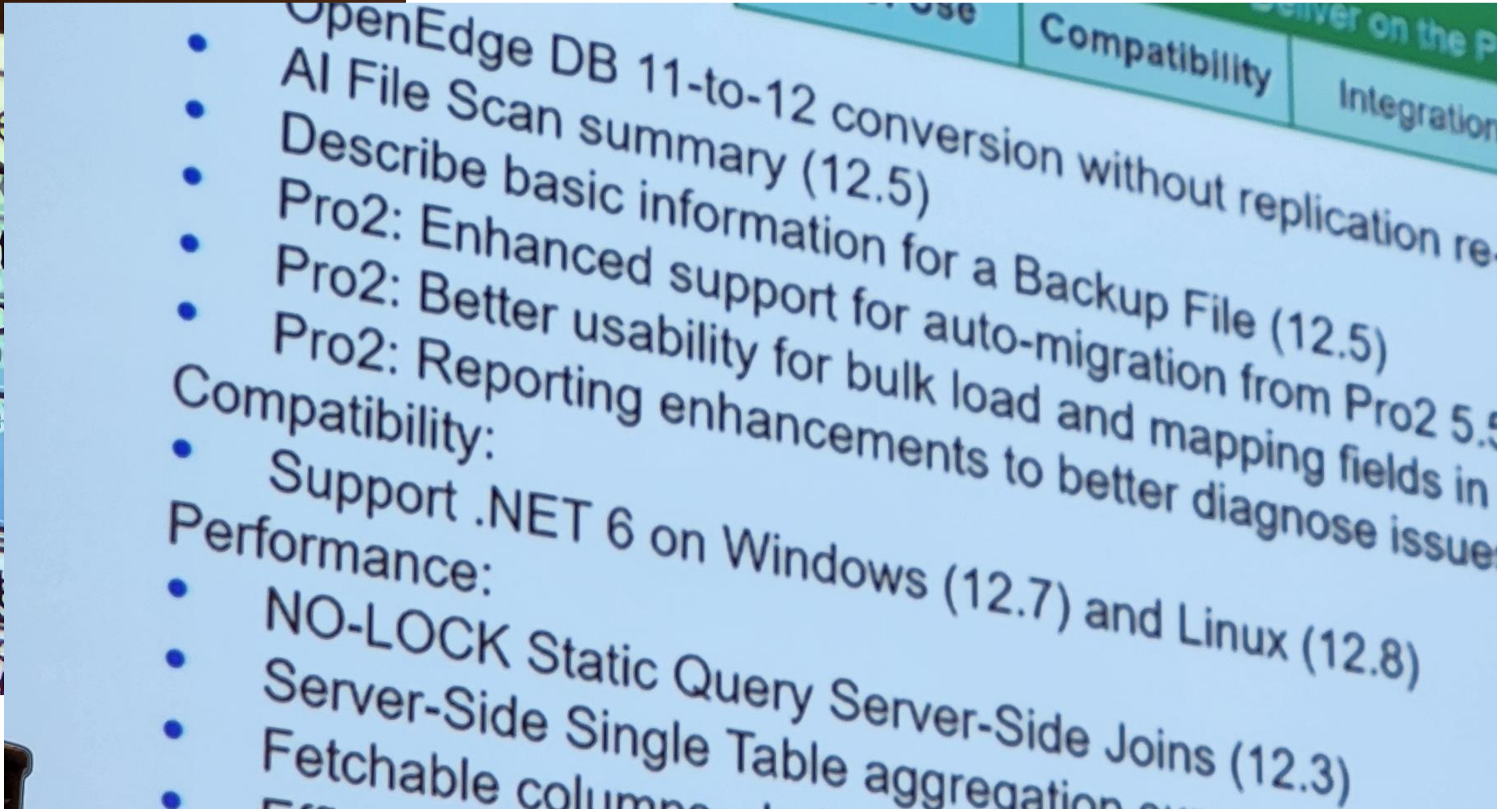
12.8 Add support f
6 for OpenEdge c

12.7 Add support fo
6 for OpenEdge
Windows

12.3 Allow init

12.3 Support fo

ind



Use
Compatibility
Deliver on the p
Integration

- OpenEdge DB 11-to-12 conversion without replication re
- AI File Scan summary (12.5)
- Describe basic information for a Backup File (12.5)
- Pro2: Enhanced support for auto-migration from Pro2 5.5
- Pro2: Better usability for bulk load and mapping fields in
- Pro2: Reporting enhancements to better diagnose issue

Compatibility:

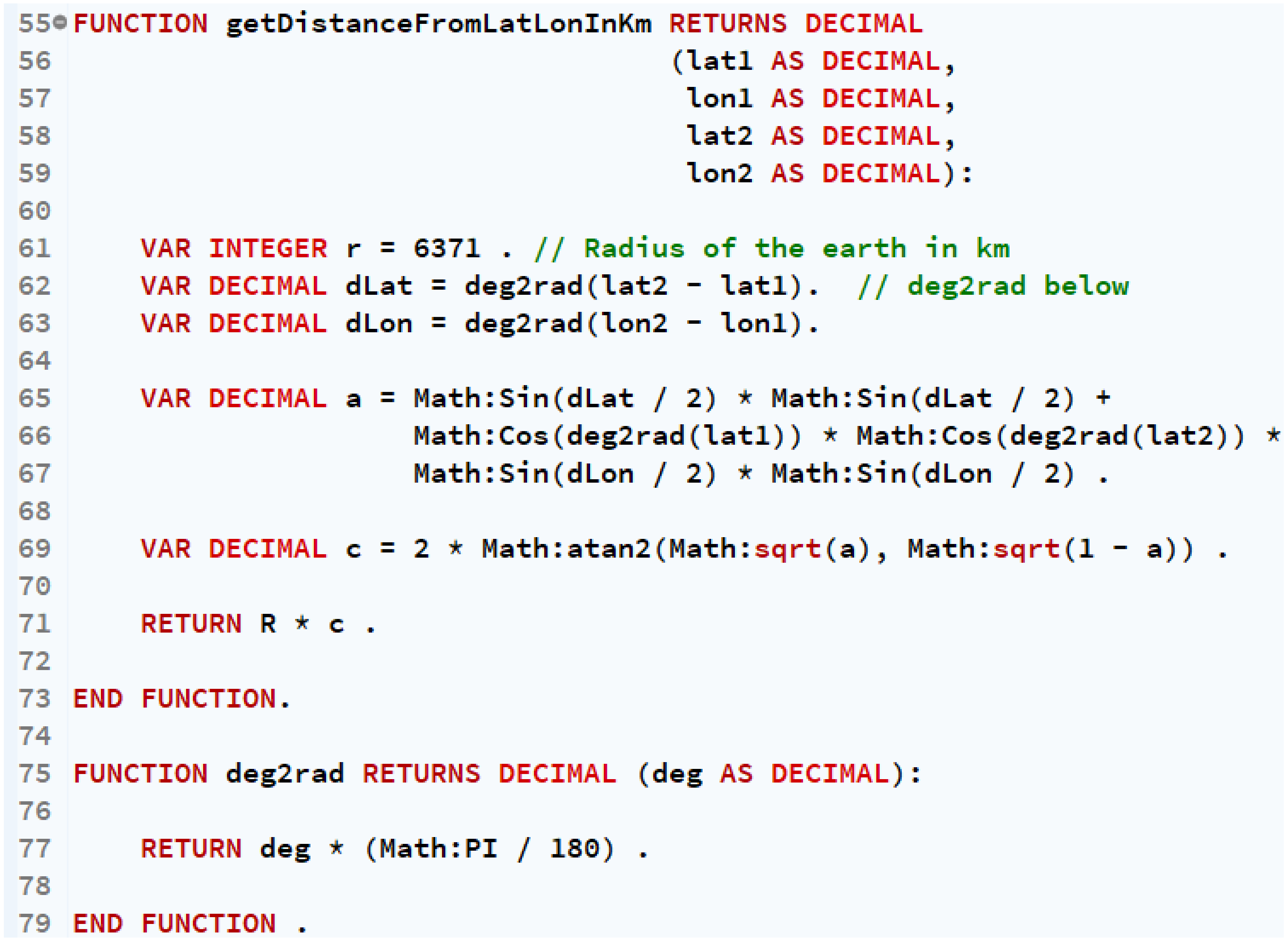
- Support .NET 6 on Windows (12.7) and Linux (12.8)

Performance:

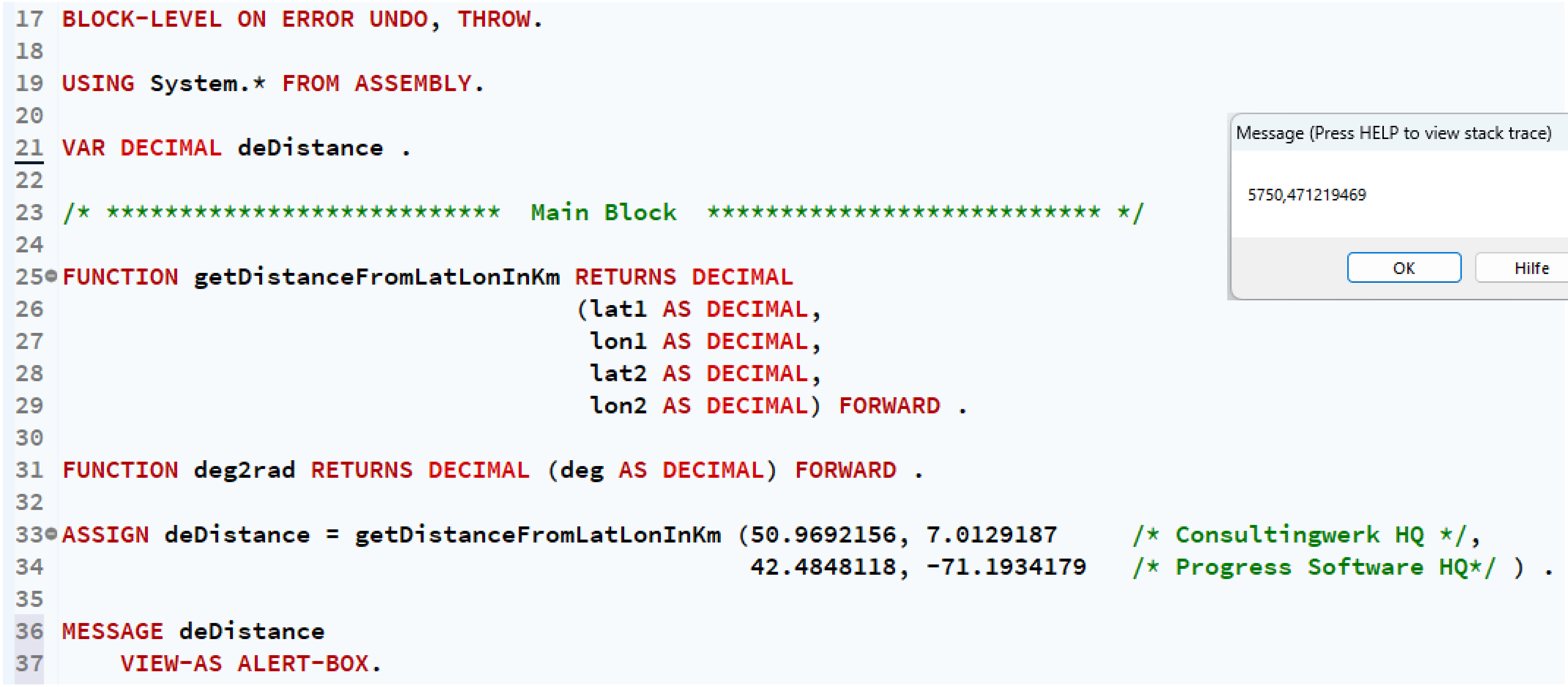
- NO-LOCK Static Query Server-Side Joins (12.3)
- Server-Side Single Table aggregation
- Fetchable column

Simple sample of non-GUI for .NET capabilities

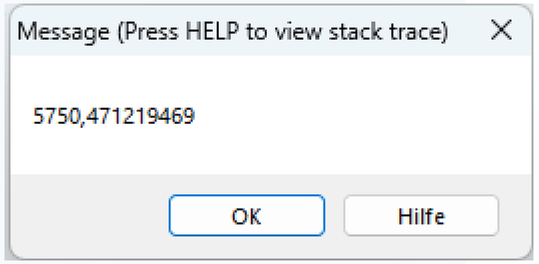
- Simple trigonometry (my 16 y/o son will hate me for this)
- Calculating the distance between Progress HQ in Burlington and Consultingwerk HQ in Cologne
- Let’s assume the world is ~~flat~~ a perfect sphere
- https://stackoverflow.com/questions/27928/calculate-distance-between-two-latitude-longitude-points-haversine-formula
- ABL lacks any trigonometrical function out of the box: SIN, COS, TAN and SQRT
- ABL lacks the constant PI (okay, it’s a constant value)
- .NET (any version) to the rescue



```
FUNCTION getDistanceFromLatLonInKm RETURNS DECIMAL
                                   (lat1 AS DECIMAL,
                                    lon1 AS DECIMAL,
                                    lat2 AS DECIMAL,
                                    lon2 AS DECIMAL):

    VAR INTEGER r = 6371 . // Radius of the earth in km
    VAR DECIMAL dLat = deg2rad(lat2 - lat1).  // deg2rad below
    VAR DECIMAL dLon = deg2rad(lon2 - lon1).

    VAR DECIMAL a = Math:Sin(dLat / 2) * Math:Sin(dLat / 2) +
                    Math:Cos(deg2rad(lat1)) * Math:Cos(deg2rad(lat2)) *
                    Math:Sin(dLon / 2) * Math:Sin(dLon / 2) .

    VAR DECIMAL c = 2 * Math:atan2(Math:sqrt(a), Math:sqrt(1 - a)) .

    RETURN R * c .

END FUNCTION.

FUNCTION deg2rad RETURNS DECIMAL (deg AS DECIMAL):

    RETURN deg * (Math:PI / 180) .

END FUNCTION .
```



```
BLOCK-LEVEL ON ERROR UNDO, THROW.

USING System.* FROM ASSEMBLY.

VAR DECIMAL deDistance .

/* ***************************  Main Block  *************************** */

FUNCTION getDistanceFromLatLonInKm RETURNS DECIMAL
                                   (lat1 AS DECIMAL,
                                    lon1 AS DECIMAL,
                                    lat2 AS DECIMAL,
                                    lon2 AS DECIMAL) FORWARD .

FUNCTION deg2rad RETURNS DECIMAL (deg AS DECIMAL) FORWARD .

ASSIGN deDistance = getDistanceFromLatLonInKm (50.9692156, 7.0129187     /* Consultingwerk HQ */,
                                               42.4848118, -71.1934179   /* Progress Software HQ*/ ) .

MESSAGE deDistance
    VIEW-AS ALERT-BOX.
```



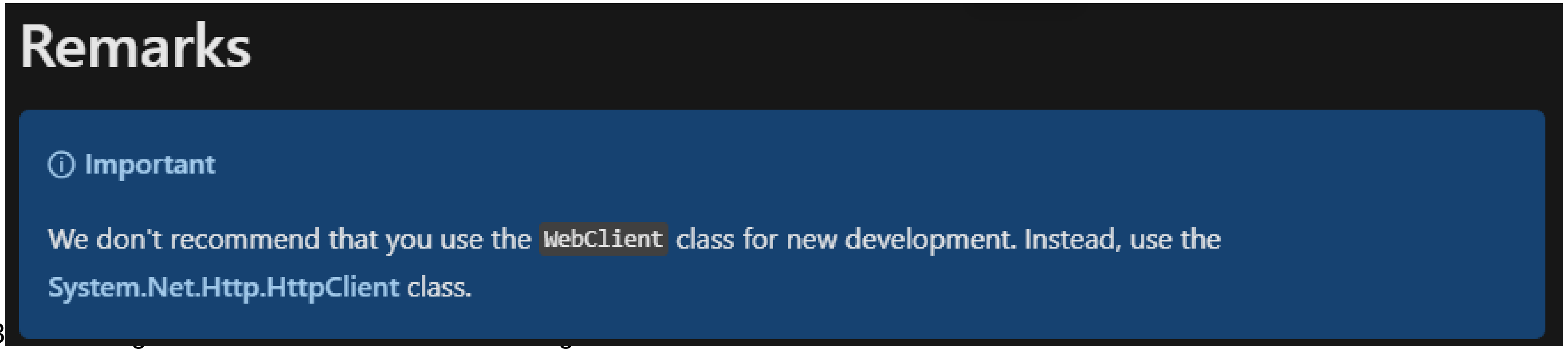
```
Message (Press HELP to view stack trace)

5750,471219469
```

Agenda

- .NET Core
- .NET and OpenEdge
- Using .NET Core in OpenEdge 12.7
- **Consuming asynchronous API's**
- Benchmarking

Consuming asynchronous API‘s

- .NET provides more and more asynchronous API’s
- Some of those changes are justified with improved performance
- For instance, Microsoft provided the System.Net.WebClient class for making HTTP requests – which is now considered obsolete
- https://learn.microsoft.com/en-us/dotnet/api/system.net.webclient?view=net-6.0



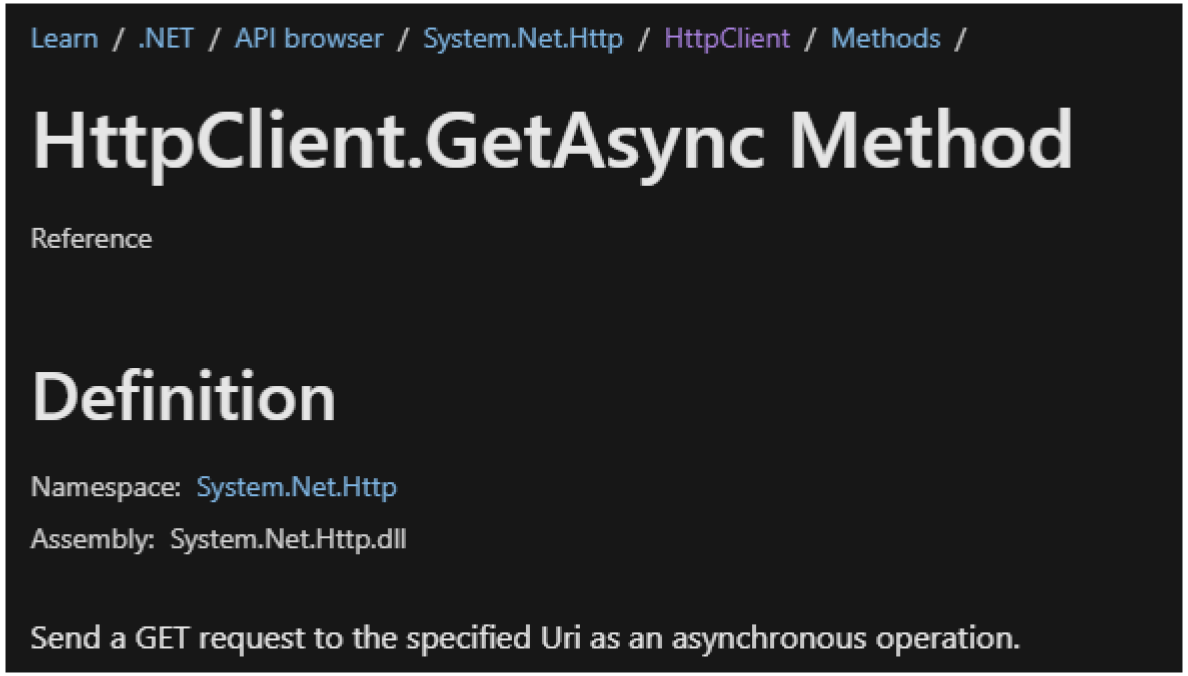
Remarks

ⓘ Important

We don't recommend that you use the `WebClient` class for new development. Instead, use the System.Net.Http.HttpClient class.

System.Net.Http.HttpClient class

- Only provides methods for asynchronous requests


Learn / .NET / API browser / System.Net.Http / HttpClient / Methods /

HttpClient.GetAsync Method

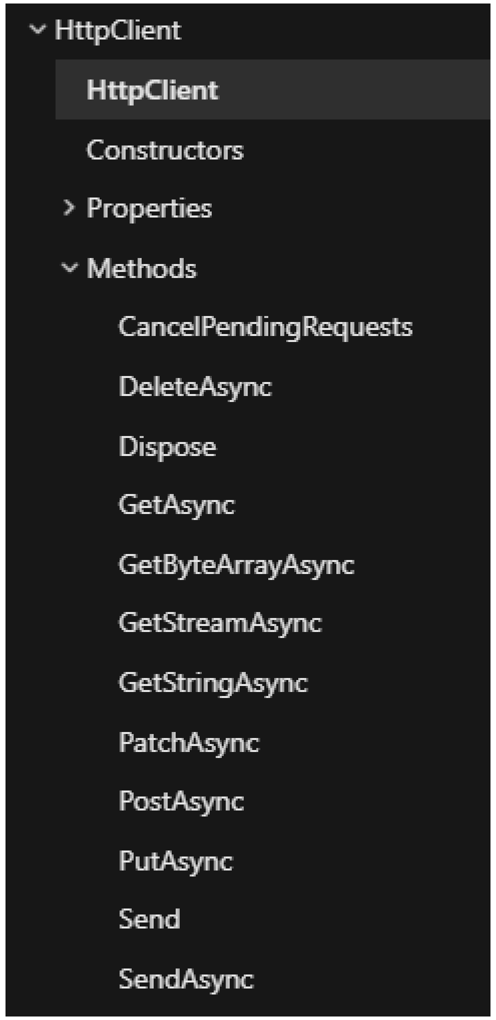
Reference

Definition

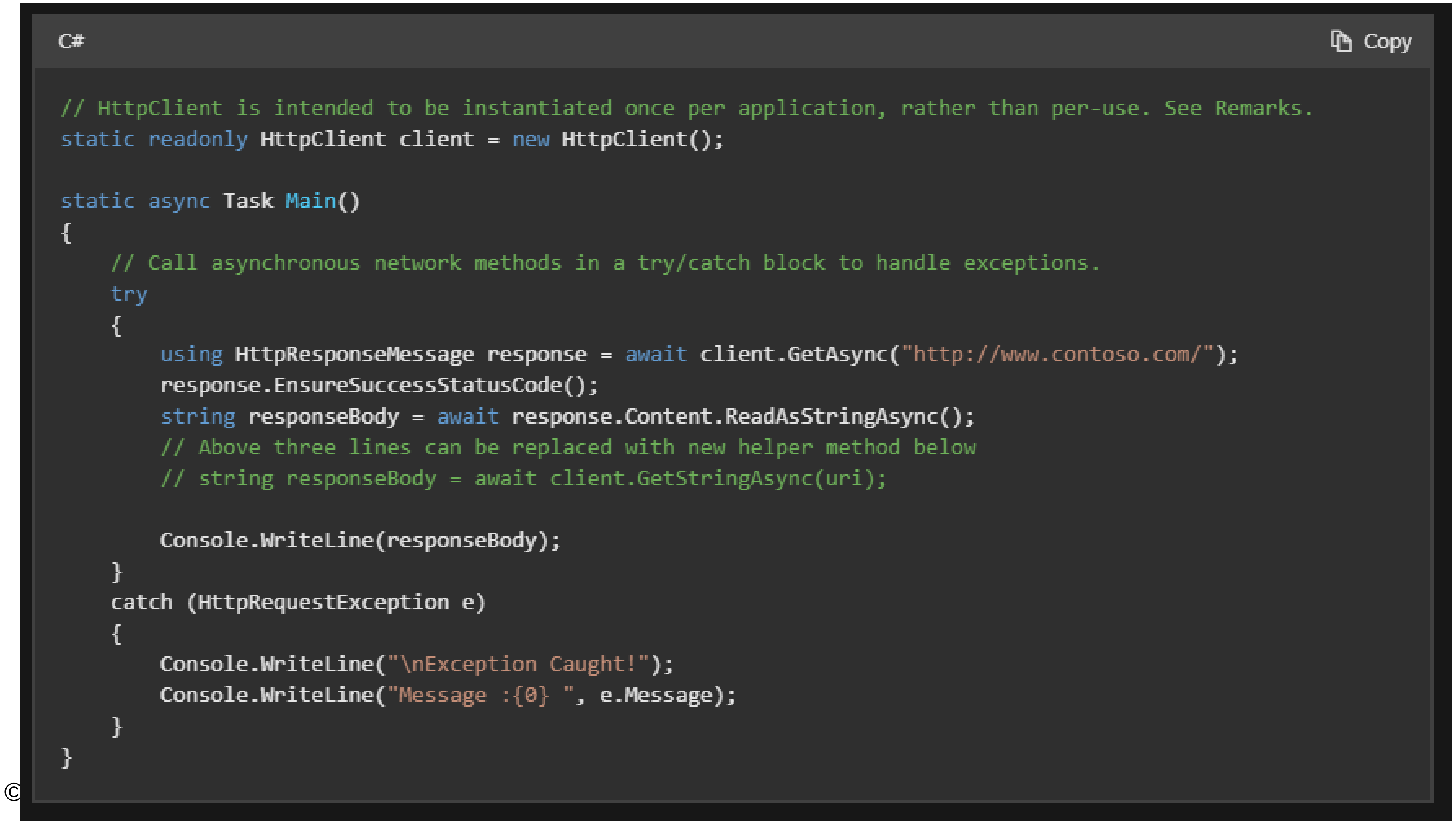
Namespace: System.Net.Http

Assembly: System.Net.Http.dll

Send a GET request to the specified Uri as an asynchronous operation.



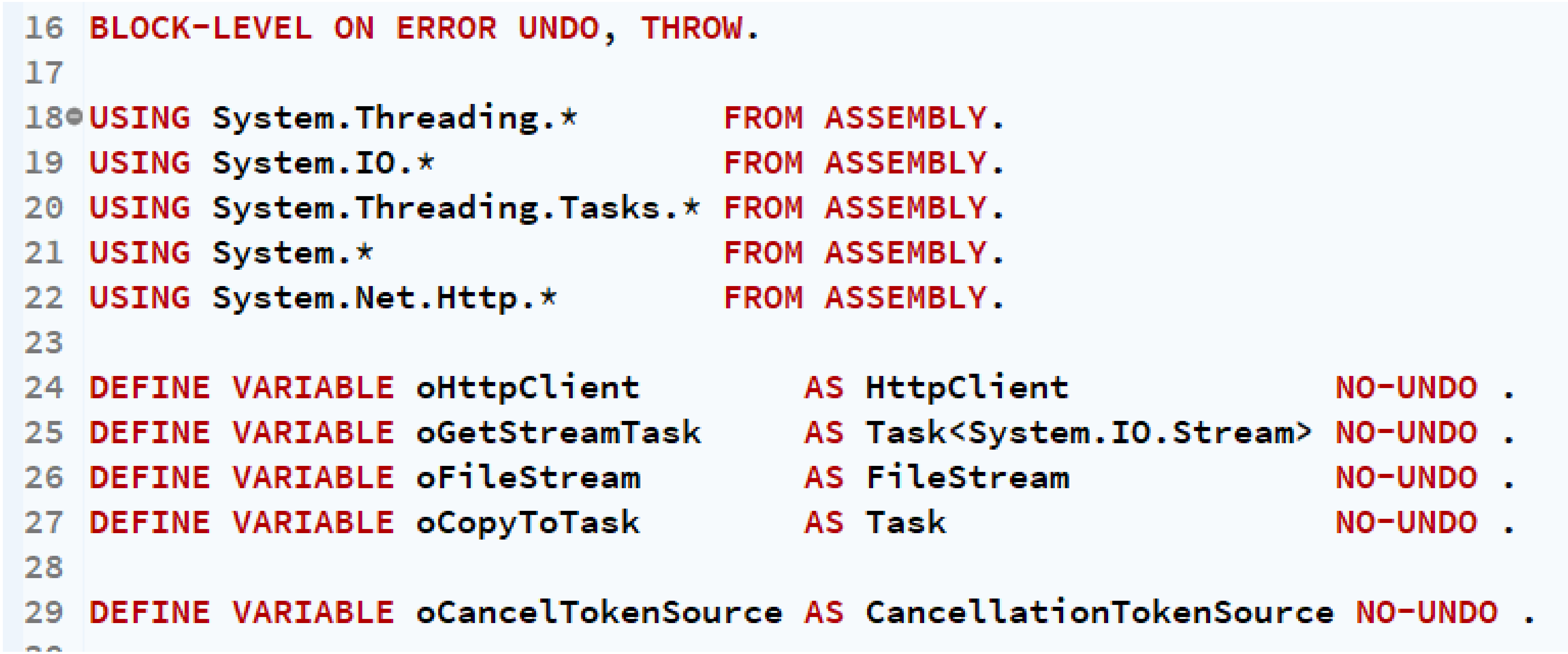
Sample usage in C#



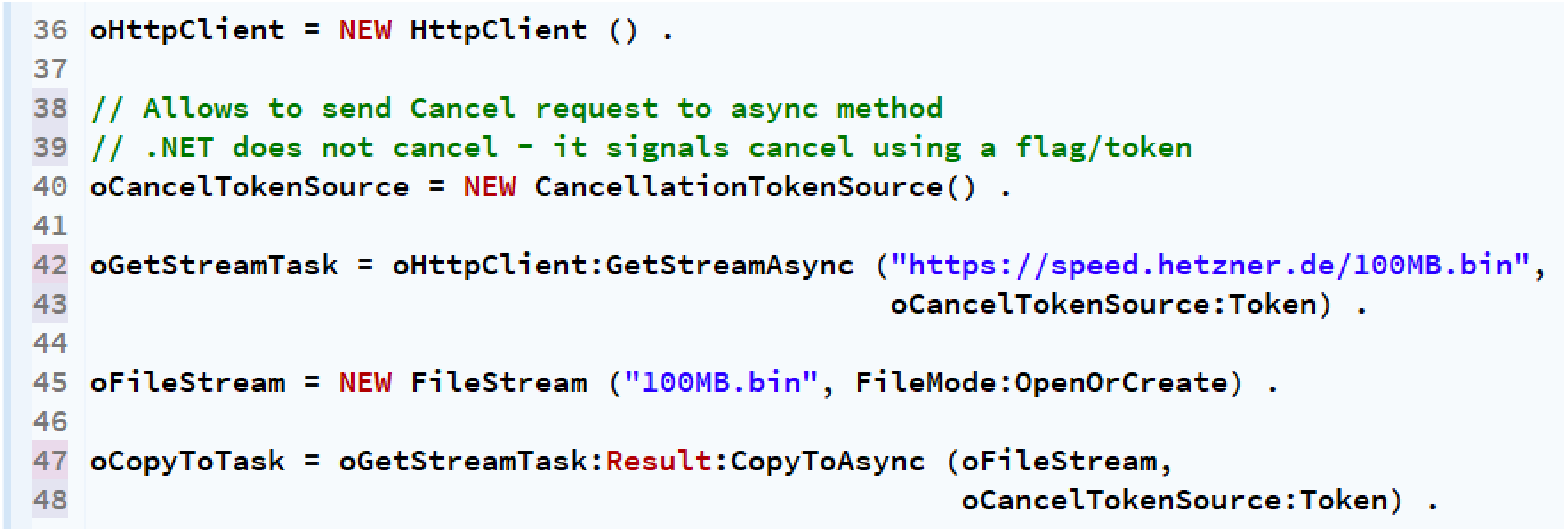
```csharp
// HttpClient is intended to be instantiated once per application, rather than per-use. See Remarks.
static readonly HttpClient client = new HttpClient();

static async Task Main()
{
    // Call asynchronous network methods in a try/catch block to handle exceptions.
    try
    {
        using HttpResponseMessage response = await client.GetAsync("http://www.contoso.com/");
        response.EnsureSuccessStatusCode();
        string responseBody = await response.Content.ReadAsStringAsync();
        // Above three lines can be replaced with new helper method below
        // string responseBody = await client.GetStringAsync(uri);

        Console.WriteLine(responseBody);
    }
    catch (HttpRequestException e)
    {
        Console.WriteLine("\nException Caught!");
        Console.WriteLine("Message :{0} ", e.Message);
    }
}
```

await operator

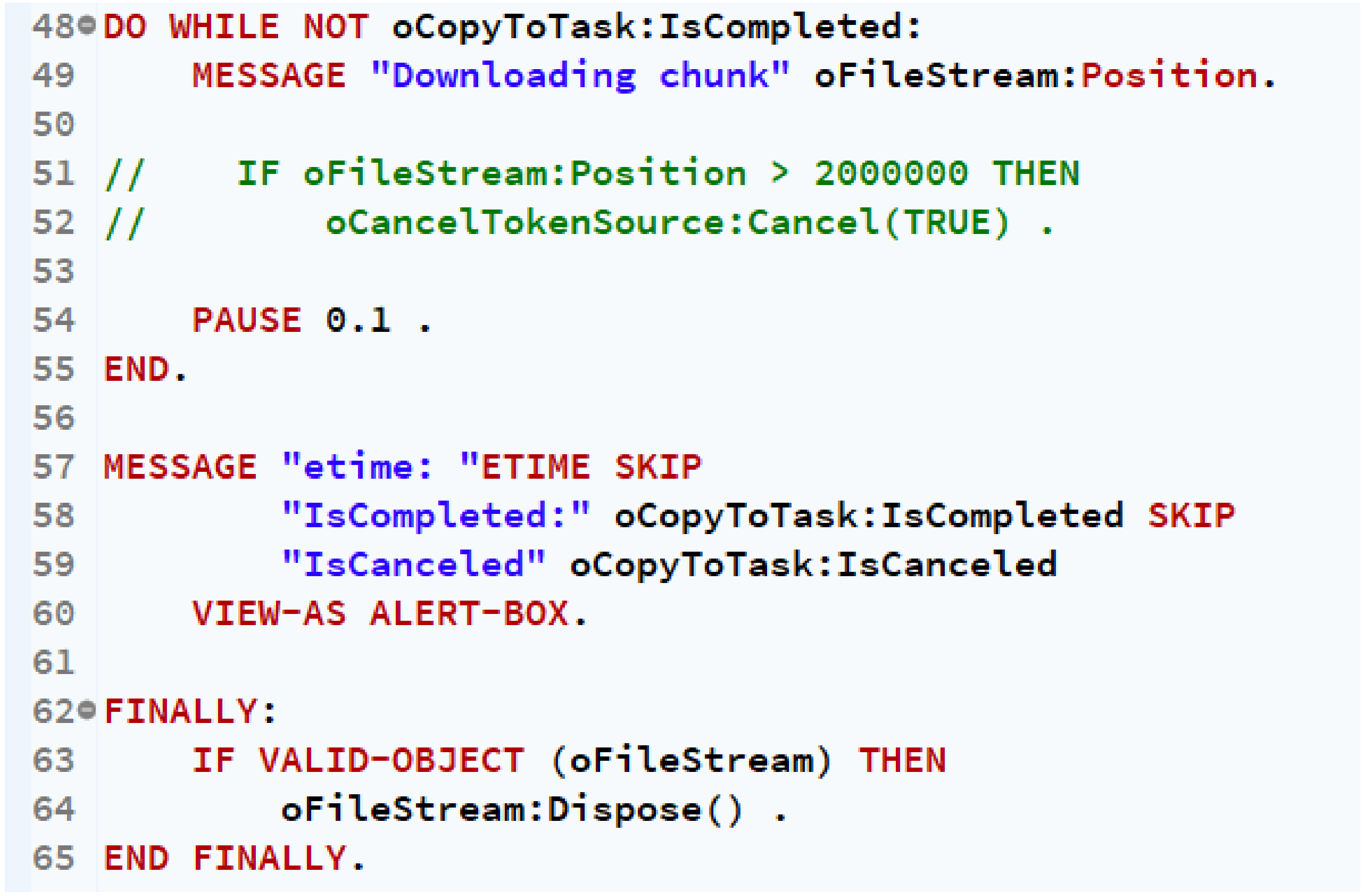
- Syntactical sugar in C#
- Waits for completion of asynchronous operation
- Like promises in JavaScript frameworks
- ABL does not provide an await operator or keyword
- GetAsync Method of the HttpClient returns an “Task” representing an asynchronous operation
- ABL can handle asynchronous operations without the await keyword by interacting with the Task instances



```
BLOCK-LEVEL ON ERROR UNDO, THROW.

USING System.Threading.*       FROM ASSEMBLY.
USING System.IO.*              FROM ASSEMBLY.
USING System.Threading.Tasks.* FROM ASSEMBLY.
USING System.*                 FROM ASSEMBLY.
USING System.Net.Http.*        FROM ASSEMBLY.

DEFINE VARIABLE oHttpClient        AS HttpClient             NO-UNDO .
DEFINE VARIABLE oGetStreamTask     AS Task<System.IO.Stream> NO-UNDO .
DEFINE VARIABLE oFileStream        AS FileStream             NO-UNDO .
DEFINE VARIABLE oCopyToTask        AS Task                   NO-UNDO .

DEFINE VARIABLE oCancelTokenSource AS CancellationTokenSource NO-UNDO .
```



```
oHttpClient = NEW HttpClient () .

// Allows to send Cancel request to async method
// .NET does not cancel - it signals cancel using a flag/token
oCancelTokenSource = NEW CancellationTokenSource() .

oGetStreamTask = oHttpClient:GetStreamAsync ("https://speed.hetzner.de/100MB.bin",
                                             oCancelTokenSource:Token) .

oFileStream = NEW FileStream ("100MB.bin", FileMode:OpenOrCreate) .

oCopyToTask = oGetStreamTask:Result:CopyToAsync (oFileStream,
                                                 oCancelTokenSource:Token) .
```



```
DO WHILE NOT oCopyToTask:IsCompleted:
    MESSAGE "Downloading chunk" oFileStream:Position.

//     IF oFileStream:Position > 2000000 THEN
//         oCancelTokenSource:Cancel(TRUE) .

    PAUSE 0.1 .
END.

MESSAGE "etime: "ETIME SKIP
        "IsCompleted:" oCopyToTask:IsCompleted SKIP
        "IsCanceled" oCopyToTask:IsCanceled
    VIEW-AS ALERT-BOX.

FINALLY:
    IF VALID-OBJECT (oFileStream) THEN
        oFileStream:Dispose() .
END FINALLY.
```

Consultingwerk
software architecture and development

Agenda

- .NET Core
- .NET and OpenEdge
- Using .NET Core in OpenEdge 12.7
- Consuming asynchronous API's
- **Benchmarking**

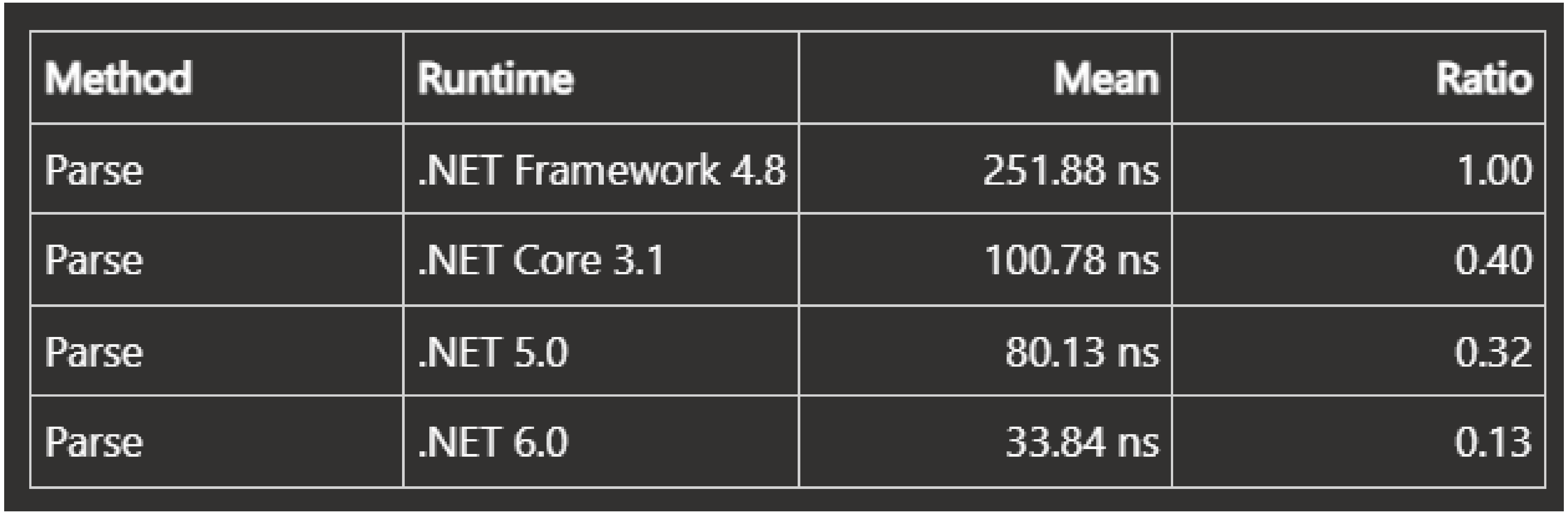
.NET Core Performance Improvements

- .NET Core has performance improvements in various areas
- Pretty generic – with huge impact and very specific
- https://devblogs.microsoft.com/dotnet/performance-improvements-in-net-6/

Benchmarking Guid:Parse()

- Method parses a GUID from String into the System.Guid type
- https://devblogs.microsoft.com/dotnet/performance-improvements-in-net-6/#system-types
- *“Let’s start with Guid. … The ability to create them quickly is important, as is the ability to quickly format and parse them. Previous releases have seen significant improvements on all these fronts, but they get even better in .NET 6”*
- *“dotnet/runtime#44918 helped avoid some overheads involved in unnecessarily accessing CultureInfo.CurrentCulture during parsing, as culture isn’t necessary or desired when parsing hexadecimal digits.”*

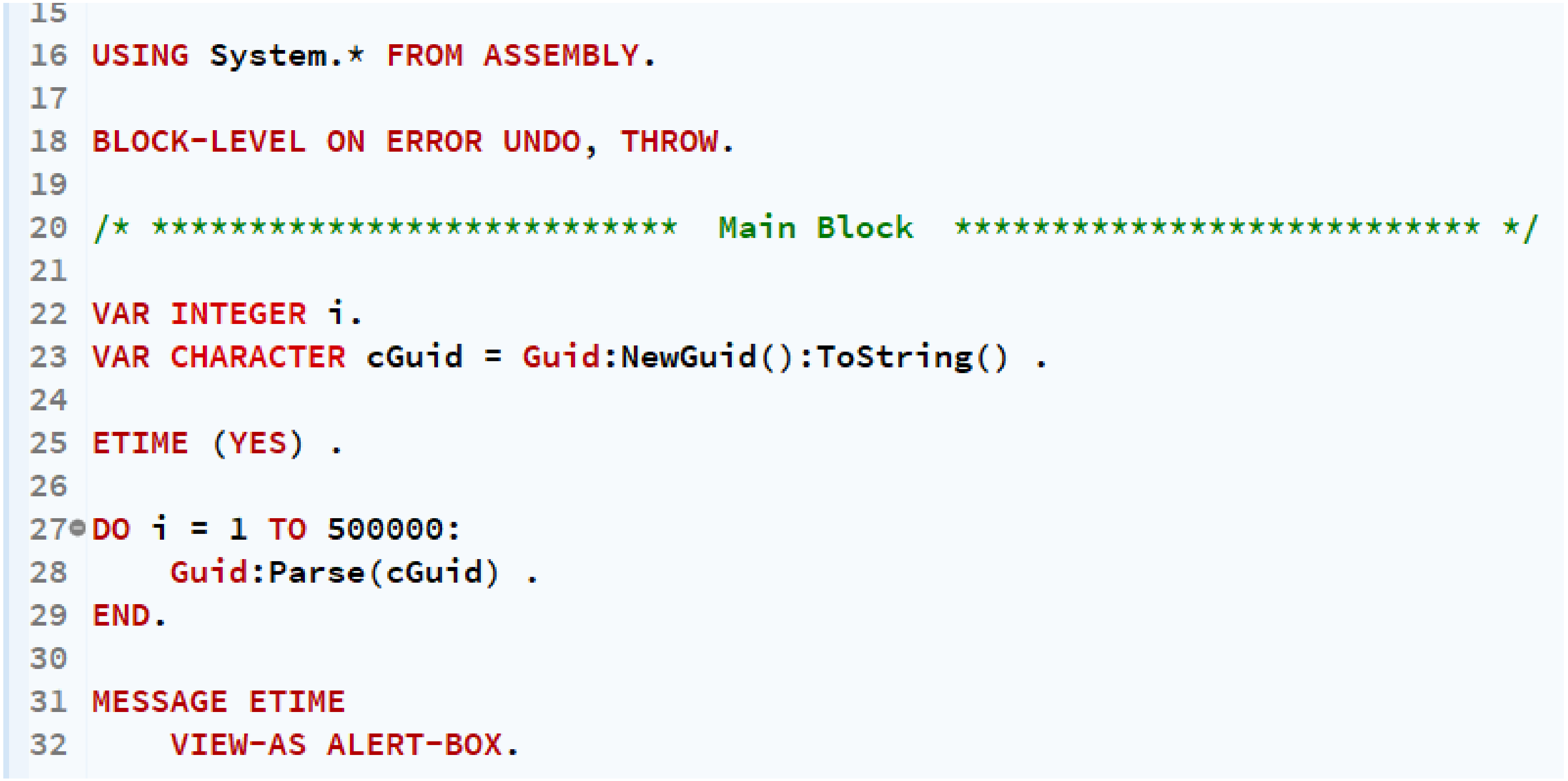
Benchmarking Guid:Parse()



Method	Runtime	Mean	Ratio
Parse	.NET Framework 4.8	251.88 ns	1.00
Parse	.NET Core 3.1	100.78 ns	0.40
Parse	.NET 5.0	80.13 ns	0.32
Parse	.NET 6.0	33.84 ns	0.13

- Remember in OpenEdge we can choose between .NET Framework 4.8 and .NET 6.0

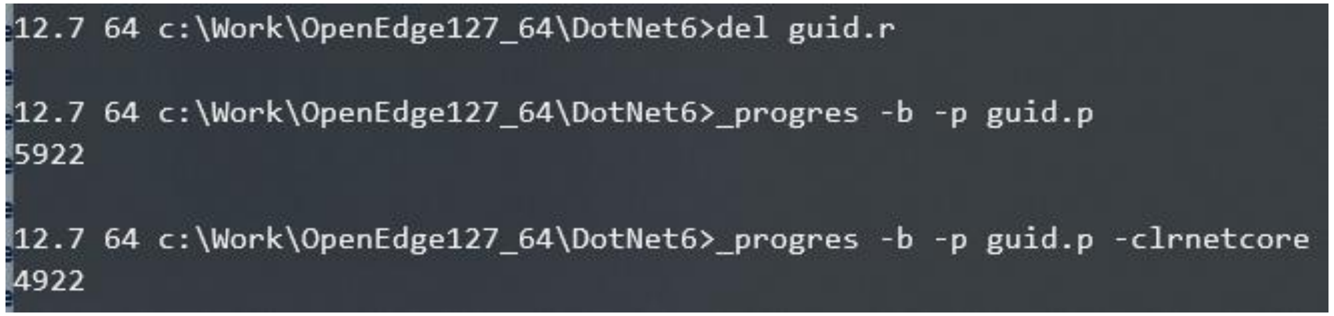
Benchmarking Guid:Parse() in ABL



```

USING System.* FROM ASSEMBLY.

BLOCK-LEVEL ON ERROR UNDO, THROW.

/* ***************************  Main Block  *************************** */

VAR INTEGER i.
VAR CHARACTER cGuid = Guid:NewGuid():ToString() .

ETIME (YES) .

DO i = 1 TO 500000:
    Guid:Parse(cGuid) .
END.

MESSAGE ETIME
    VIEW-AS ALERT-BOX.
```

Benchmarking Guid:Parse() in ABL

- We see ~ 5 seconds (.NET 6) vs. ~ 6 seconds (.NET Framework 4.8)
- We‘re not seeing the 7 times faster as the previous slide showed
- We’re still running an ABL loop 500.000 times – that has overhead
- We’re facing some more overhead on the bridge (500.000 times CHARACTER to System.String conversion, dynamic invocation)
- But .NET 6.0 sticks to the promise of being faster



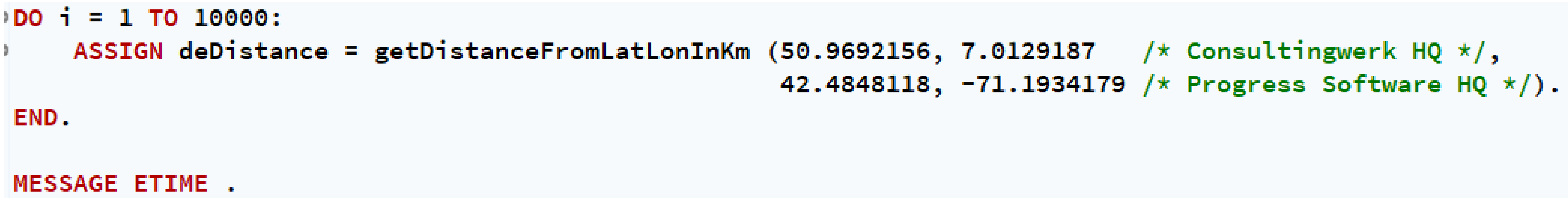
```
12.7 64 c:\Work\OpenEdge127_64\DotNet6>del guid.r

12.7 64 c:\Work\OpenEdge127_64\DotNet6>_progres -b -p guid.p
5922

12.7 64 c:\Work\OpenEdge127_64\DotNet6>_progres -b -p guid.p -clrnetcore
4922
```

Let‘s do the math …

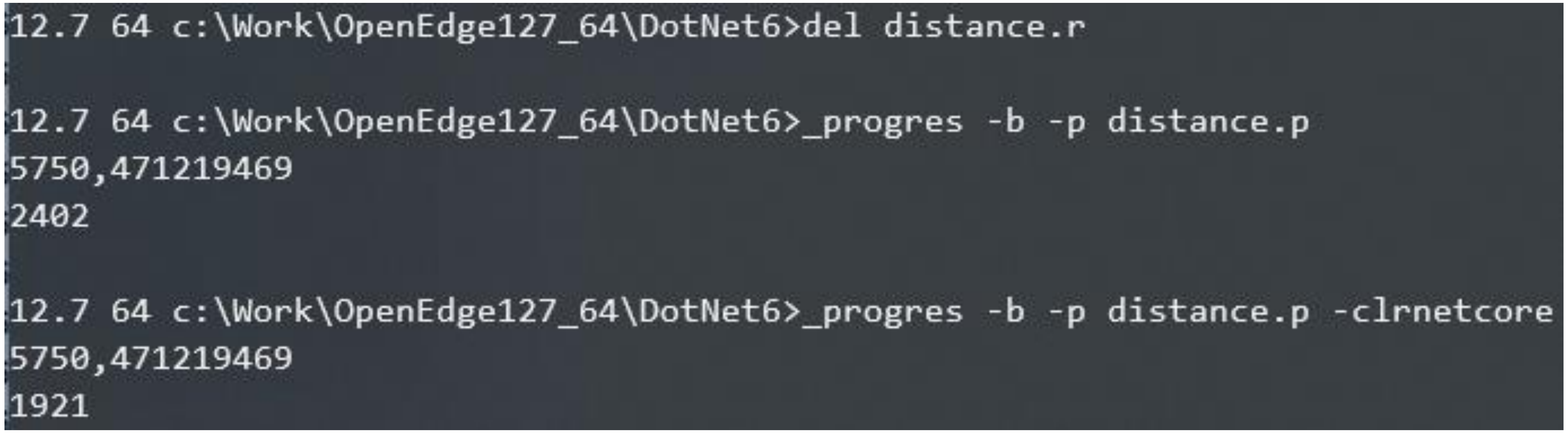
- .NET Framework 4.8 vs .NET 6.0 was a difference of ~ 220 nano seconds
- **220 nano seconds times 500.000 = ~ 0.115 seconds**
- So there’s an additional .8 seconds gained somewhere else – general infrastructure? The bridge?
- YMWV! Benchmark from blog article and my own benchmark in ABL done on different systems. But I doubt that that will explain the .8 seconds alone

Consultingwerk
software architecture and development

Benchmarking GetDistanceFromLatLonInKm in ABL



```
DO i = 1 TO 10000:
    ASSIGN deDistance = getDistanceFromLatLonInKm (50.9692156, 7.0129187   /* Consultingwerk HQ */,
                                                   42.4848118, -71.1934179 /* Progress Software HQ */).
END.

MESSAGE ETIME .
```



```
12.7 64 c:\Work\OpenEdge127_64\DotNet6>del distance.r

12.7 64 c:\Work\OpenEdge127_64\DotNet6>_progres -b -p distance.p
5750,471219469
2402

12.7 64 c:\Work\OpenEdge127_64\DotNet6>_progres -b -p distance.p -clrnetcore
5750,471219469
1921
```

Benchmarking GetDistanceFromLatLonInKm in ABL

- Improvements .NET Framework vs. .NET Core are 2.4 seconds to 1.9 seconds
- That‘s ~ 20%
- Whoever is responsible for that: Wow!

Questions

Consultingwerk

software architecture and development