Nützliche Links

Montag, 28. Oktober 2024 08:49

Microsoft Doku

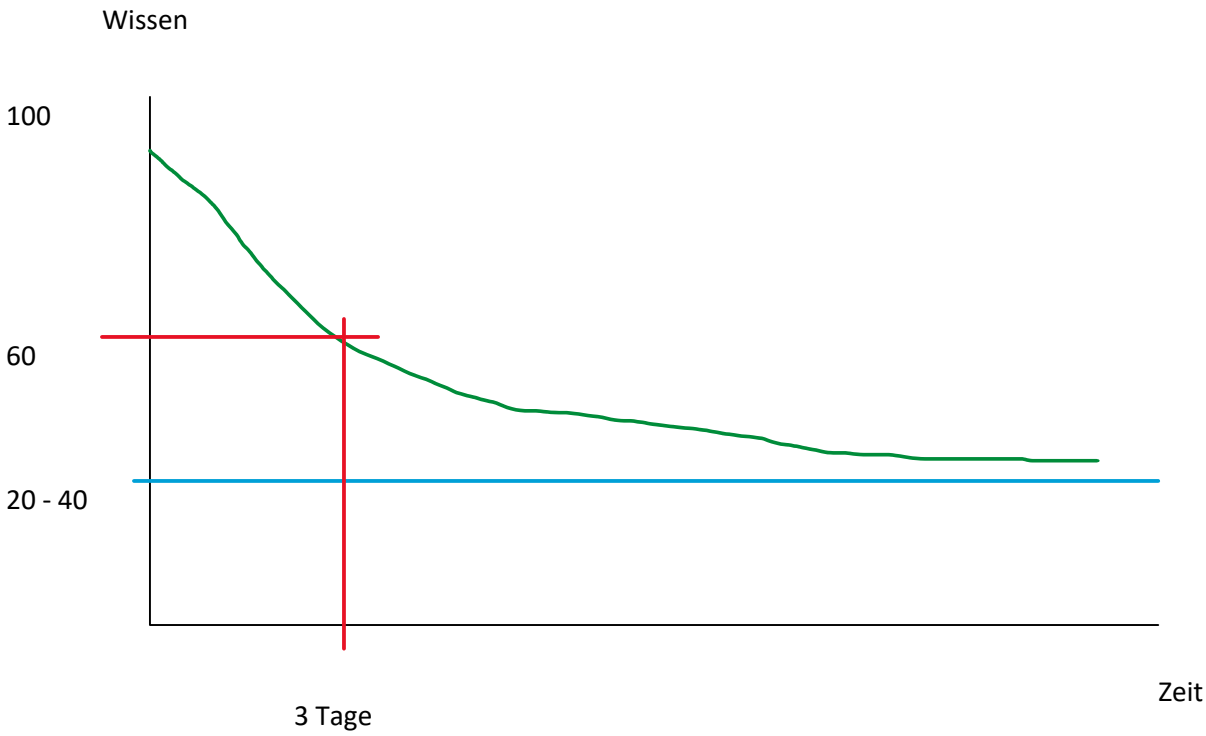
https://docs.microsoft.com/
https://learn.microsoft.com/en-us/dotnet/ (.NET)

.NET Quellcode
https://source.dot.net

Bücher
https://www.it-visions.de/buecher/Default.aspx

Vergessen

Montag, 28. Oktober 2024 09:53



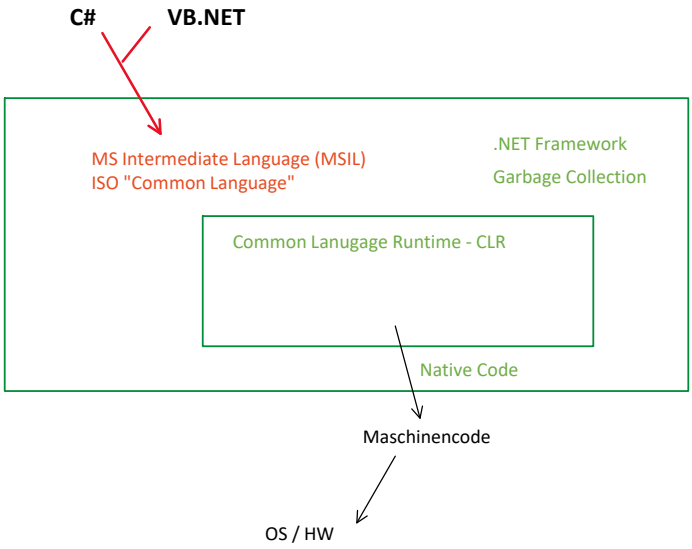
Workshop

Montag, 28. Oktober 2024 10:55

Inhalte des Auftakt-Workshops, siehe Unterseiten

.NET Framework

Montag, 2. September 2024 08:35



Objektorientierung

Versch. Technologien

Speicherverwaltung

.

C → C++ → C#

```
// C
i = i + 1;

// C++
i++;

// C#
i += 1;
```

Kurze(!) Geschichte des .NET Framework

Montag, 2. September 2024 09:58

.NET Framework (".NET Classic")

Jahr	Version	Bemerkung
2000	(1.0) 1.1	Visual Studio 2000 - ASP.NET WebForms
2003	2.0	600 Namesräume - ASP.NET → "IIS Technologies"
2007	3.0	~~WPF, EntityFramework, LINQ~~
2008	3.5	WPF, EntityFramework, LINQ (3.5 SP1)
2010	4.0	TPL, TCP/IP
2012	4.5	Roslyn-Compiler - 14.000 Namensräume
	...	
2016	4.8	"Final Version"

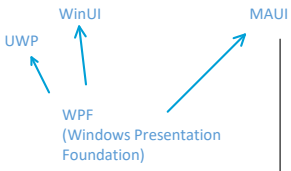
.NET (Core)

- leichtgewichtig
- plattformunabhängig (jetzt wirklich)
- Open Source

Jahr (Nov.)	Version	Bemerkungen
2016	1.0	Backend only - keine UI! - 850 Namensräume → nuget
	...	
2019	3.0	Windows Forms, WPF (Windows only) - ~~Open Source~~
2020	5.0	98% .NET Framework - .NET ~~Core~~ - Standard Term Support (STS)
2021	6.0	Long Term Support (LTS)
2022	7.0	STS
2023	8.0	LTS

Mehrschicht-Architektur

Montag, 2. September 2024 11:05



MAUI

Blazor

ASP.NET MVC

Desktop: Executable - .exe
Web: Webanwendung - .dll

UI

Windows Forms

HTML 5 (CSS, JavaScript)

UnitTesting

Klassenbibliothek(en) - .dll

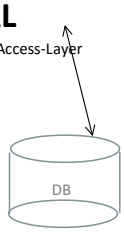
BL

C#
nuget

DAL

Data-Access-Layer

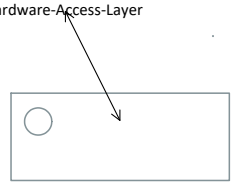
ADO.NET
Entity Framework



HAL

Hardware-Access-Layer

C#
Drittanbieter-Bibliotheken



Klassenbibliothek(en) - .dll

Tooling

Montag, 2. September 2024 11:53

Basis

Editor + Windows SDK

Visual Studio

Visual Studio Code - komplett kostenlos, komplett in .NET entwickelt
Visual Studio Community Edition - kostenlos, aber kein kommerzieller Einsatz!
Visual Studio Professional
Visual Studio Enterprise

Externe Tools

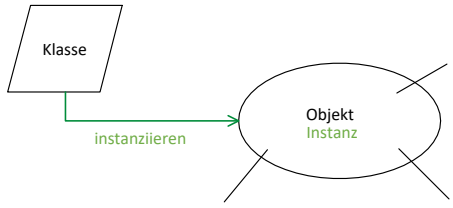
Github Copilot
Git - Quellcode-Verwaltung, kostenlos
DBeaver - DB-Management, Open Source

Objektorientierte Programmierung

Montag, 28. Oktober 2024 13:43

Paradigmen

- "Separation of Concerns"
  - Trennung und Zuordnung von Zuständigkeiten
- Kapselung
  - Interne Vorgängen bleiben intern
- Wieder-/Weiterverwendung
  - Vererbung
  - Überladung



Ereignisse

Konzept: "Bescheid sagen"
→ IoC: Inversion of control

Eigenschaften
(Properties)

beschreiben eine(!) Eigenschaft des Objekts

- Hauptwörter
- selten EIgenschaftswörter

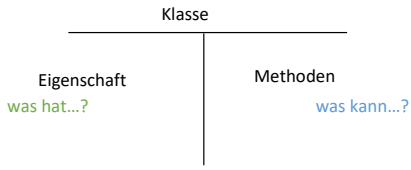
Methoden

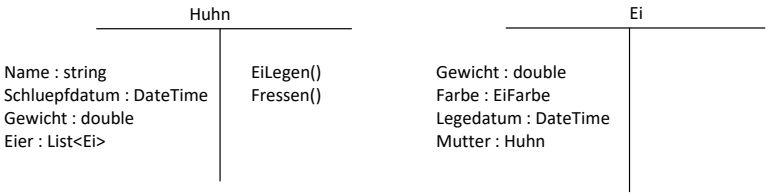
machen etwas mit dem/für das Objekt

- Verben (Tu-Wörter)

Objektmodell Eierfarm

Montag, 28. Oktober 2024 14:13





Nützliche Tastenkombinationen(*)

Montag, 28. Oktober 2024 14:48

* Alle Angaben C#-Einstellungen
(Extras/Einstellungen importieren…)

Visual Studio

Code bearbeiten

Auskommentieren	Strg+#	Strg+K, Strg+C
Einkommentieren	Strg+#	Strg+K, Strg+U
SmartTag-Menü öffnen	Strg+. (weitere Auswahl Pfeiltasten)	
Code "schön machen"	Strg+K, Strg+D	

Code navigieren

Gehe zu Definition/Deklaration	F12
Zurück zur letzten Position	Strg+-

Debuggen

Einzelschritt	F11
Prozedurschritt	F10
Verlassen	Shift+F11
Haltepunkt umschalten	F9

C#-Syntax

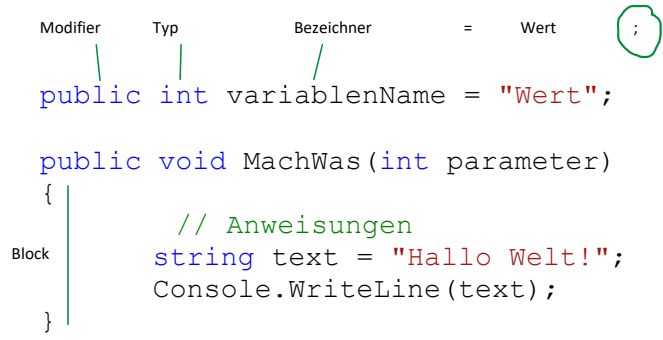
Montag, 28. Oktober 2024 15:27

```
int a = 12;
int A = 13;
```

Groß-Kleinschreibung

→ C# ist case sensitive!

- Nutzen: Lesbarer Code
- Coding Richtlinie:
Alle öffentlichen Member (Methoden, Eigenschaften Klassen, Typen)	Groß
Alle lokalen Elemente (auch Übergabeparameter)	klein



Schleifen in C#

Dienstag, 29. Oktober 2024 08:51

```
for (int i=0; i < 10; i++)
{
      Console.WriteLine($"{i * 10}");
}

Console.WriteLine($"{i} Durchläufe!");
```

"Strings are immutable!" - Strings sind nicht änderbar!

Dienstag, 29. Oktober 2024 08:56

```
string hallo = "Hallo";
string welt = "Welt!";

string meldung1 = hallo + " " + welt;

// Seit .NET 2.0:
StringBuilder builder = new StringBuilder(hallo);
builder.Append(" ");
builder.Append(welt);
string meldung2 = builder.ToString();

string meldung3 = String.Format("{0} {1}", hallo, welt);

double zahl = 3.1415926536;
string pi = String.Format("{0:#,##0.0000}", zahl); // 3.1415


// Seit .NET 4.0:
string meldung4 = $"{hallo} {welt}";
string p2 = $"{zahl:0.000}";
```

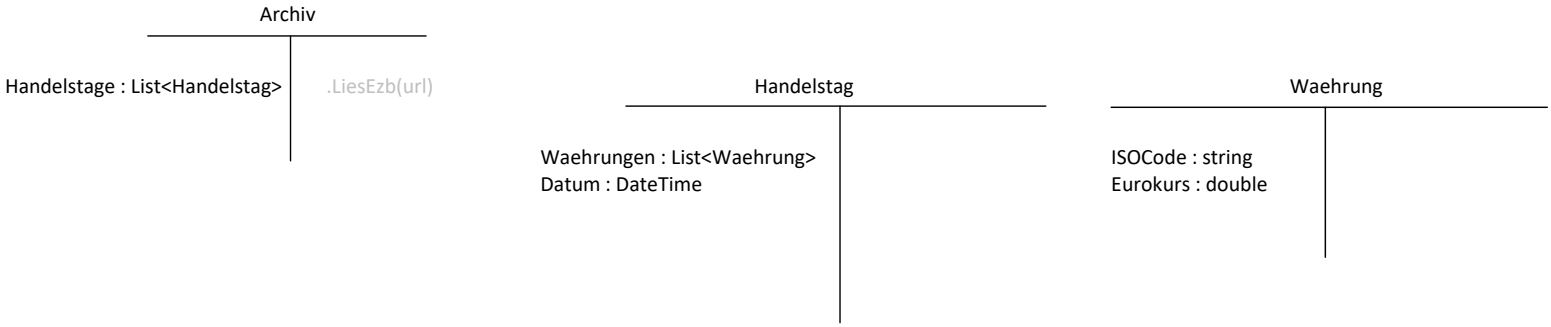
XML

Dienstag, 29. Oktober 2024 09:42

```xml
<!-- XML besteht aus Tags -->
<Tag>Content</Tag>

<!-- Content kann wieder XML sein -->
<Tag>
      <Tag>Content</Tag>
</Tag>

<!-- Attribute in XML -->
<Tag Attribut1="Wert1" Attribut2="Wert2">Content</Tag>

<!-- Singuläre Tags (ohne Content) -->
<Tag />

<!-- Singuläre Tags mit Attributen -->
<Tag Attribut1="Wert1" Attribut2="Wert2" />
```

Objektmodell EZB-Wechselkurse

Dienstag, 29. Oktober 2024 10:26



Nützliche Code-Snippets

Dienstag, 29. Oktober 2024 11:43

Properties

prop	Auto-Property
propfull	Full qualified Property (mit Backing Field)

Konstruktor

ctor	Standardkonstruktor

Nullable in C#

Dienstag, 29. Oktober 2024 11:47

Datentypen

```
// Fehler hier
int zahl = GetZahlFromDb(...);

// Nullable
System.Nullable<int> zahl1 = GetZahlFromDb(...);
int? zahl2 = GetZahlFromDb(...);

// Verwendung
int a = 12;
int? b = null;

int c1 = a + b; // Fehler
int c2;
if (b.HasValue)
{
        c2 = a + b.Value;
}

// VB.NET IIf(...)
int c3 = a + (b.HasValue ? b.Value : -1);
int c4 = a + (b ?? -1);
```

Komplexe Typen

```
Radio radio = null!;

// Fehler, weil radio null
radio.Sender = "1live";

// Wieso dann nullable types?
#nullable disable
            Handelstag handelstag2 = null!;

            Handelstag? handelstag1 = null;

            // Warning hier
            handelstag1.Datum = DateTime.UtcNow;

            if (handelstag1 != null)
            {
                // keine Warning
                handelstag1.Datum = DateTime.UtcNow;
            }


            Handelstag handelstag3 = null!;

            if (0.5 % 2 == 2)
            {
                handelstag3 = new Handelstag();
            }
#nullable enable
```

Language INtegrated Query

Dienstag, 29. Oktober 2024 13:58

SQL

```
SELECT * FROM Customers WHERE City LIKE 'Witten%'
```

C# ohne LINQ

```
List<Customer> customers = GetCustomers();

List<Customer> wittener = GetCustomerByCity("Witten");


// Funktion zur Ermittlung aller Customer einer City
List<Customer> GetCustomerByCity(string city)
{
        List<Customer> found=new List<Customer>();

        foreach (Customer cu in customers)
        {
                if (cu.City.StartsWith(city))
                {
                        found.Add(cu)
                }
        }

        return found;
}
```

C# mit LINQ

```
List<Customer> customers = GetCustomers();

// Deklarative Syntax
var q1 = from cu in customers
                        where cu.City.StartsWith("Witten")
                        select cu;

// Methoden-Syntax (Lambda-Syntax)
var q2 = customers.Where(cu => cu.City.StartsWith("Witten"));


// Deferred Execution
// Erst bei Zugriff auf die Ergebnismenge
// wird Query ausgeführt
List<Customer> wittener = q1.ToList();

foreach (Customer cu in q2)
{
        //...
```

IEnumerable<Customer>

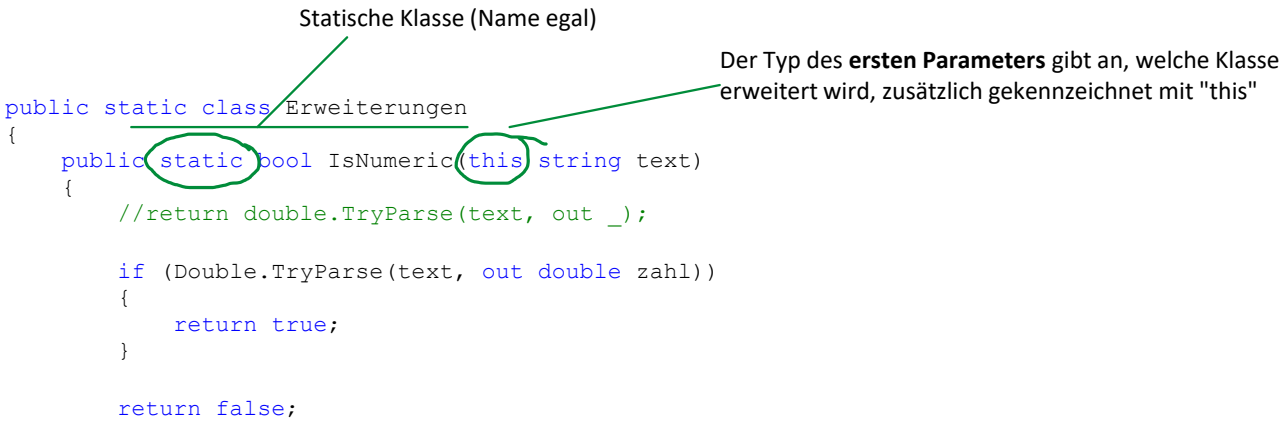
SELECT

```
// Anonymer Typ - Compiler erzeugt neue Klasse (ohne Name)
// Deklarative Syntax
var q1 = from cu in customers
                        where cu.City.StartsWith("Witten")
                        select new { cu.Vorname, cu.Nachname };

// Methoden-Syntax (Lambda-Syntax)
var q2 = customers.Where(cu => cu.City.StartsWith("Witten"))
                                        .Select(cu => new { cu.Vorname, cu.Nachname });

// Projektion - Erzeugung von Instanzen einer vorh. Klasse aus dem Query-Ergebnis
// Deklarative Syntax
var q1 = from cu in customers
                        where cu.City.StartsWith("Witten")
                        select new DisplayCustomer(cu.Vorname, cu.Nachname);

// Methoden-Syntax (Lambda-Syntax)
var q2 = customers.Where(cu => cu.City.StartsWith("Witten"))
                                        .Select(cu => new DisplayCustomer(cu.Vorname, cu.Nachname));
```

Selbstgeschriebene Klassse

Erweiterungsmethoden

Mittwoch, 30. Oktober 2024 09:28




```
public static class Erweiterungen
{
    public static bool IsNumeric(this string text)
    {
        //return double.TryParse(text, out _);

        if (Double.TryParse(text, out double zahl))
        {
            return true;
        }

        return false;
```

Variablen/Objekte (Instanzen) im Arbeitsspeicher

Mittwoch, 30. Oktober 2024 13:41