무료 전자 책

배우기

.NET Framework

Free unaffiliated eBook created from
Stack Overflow contributors.

#.net

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: -net-framework

It is an unofficial and free .NET Framework ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official .NET Framework.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1: .NET Framework

.NET Framework Microsoft . .NET MSIL (Microsoft Intermediate Language) . MSIL (CLR) .

.NET Framework "Hello World" . "Hello World" "Hello World" . . , .

.NET

.

1.0	2002-02-13
1.1	2003-04-24
2.0	2005-11-07
3.0	2006-11-06
3.5	2007-11-19
3.5 SP1	2008-08-11
4.0	2010-04-12
4.5	2012-08-15
4.5.1	2013-10-17
4.5.2	2014-05-05
4.6	2015-07-20
4.6.1	2015-11-17
4.6.2	2016-08-02
4.7	2017-04-05

Compact Framework

1.0	2000-01-01
2.0	2005-10-01

3.5	2007-11-19
3.7	2009-01-01
3.9	2013-06-01

4.2	2011-10-04
4.3	2012-12-04
4.4	2015-10-20

Examples

C # Hello World

```
using System;

class Program
{
    // The Main() function is the first function to be executed in a program
    static void Main()
    {
        // Write the string "Hello World to the standard out
        Console.WriteLine("Hello World");
    }
}
```

`Console.WriteLine` . "Hello World" "Hello World" . `.ToString` . .NET Framework .

.NET Fiddle

C #

Visual Basic .NET Hello World

```
Imports System

Module Program
    Public Sub Main()
        Console.WriteLine("Hello World")
    End Sub
End Module
```

.NET Fiddle

Visual Basic .NET

Hello World in F

```
open System

[<EntryPoint>]
let main argv =
    printfn "Hello World"
    0
```

.NET Fiddle

F #

C ++ / CLI Hello World

```
using namespace System;

int main(array<String^>^ args)
{
    Console::WriteLine("Hello World");
}
```

PowerShell Hello World

```
Write-Host "Hello World"
```

PowerShell

Nemerle

```
System.Console.WriteLine("Hello World");
```

Oxygene

```
namespace HelloWorld;

interface

type
  App = class
  public
    class method Main(args: array of String);
  end;

implementation

class method App.Main(args: array of String);
begin
  Console.WriteLine('Hello World');
end;

end.
```

Boo Hello World

```
print "Hello World"
```

Hello World in Python (IronPython)

```
print "Hello World"
```

```
import clr
from System import Console
Console.WriteLine("Hello World")
```

IL

```
.class public auto ansi beforefieldinit Program
       extends [mscorlib]System.Object
{
  .method public hidebysig static void  Main() cil managed
  {
    .maxstack  8
    IL_0000:  nop
    IL_0001:  ldstr      "Hello World"
    IL_0006:  call       void [mscorlib]System.Console::WriteLine(string)
    IL_000b:  nop
    IL_000c:  ret
  }

  .method public hidebysig specialname rtspecialname
          instance void  .ctor() cil managed
  {
    .maxstack  8
    IL_0000:  ldarg.0
    IL_0001:  call       instance void [mscorlib]System.Object::.ctor()
    IL_0006:  ret
  }

}
```

.NET Framework : https://riptutorial.com/ko/dot-net/topic/14/-net-framework-

2: .NET

.NET Core Microsoft GitHub .NET . Windows, macOS Linux , / IoT .

.NET Core (, , ,) .

Microsoft .

.NET Core , ( ) . .NET Core .

- ASP.NET
- Windows 10 Windows (UWP)
- Xamarin.Forms

.NET Core .NET .NET .

.NET .NET .NET API API . .NET .NET .NET .

Examples

```
public class Program
{
    public static void Main(string[] args)
    {
        Console.WriteLine("\nWhat is your name? ");
        var name = Console.ReadLine();
        var date = DateTime.Now;
        Console.WriteLine("\nHello, {0}, on {1:d} at {1:t}", name, date);
        Console.Write("\nPress any key to exit...");
        Console.ReadKey(true);
    }
}
```

.NET : https://riptutorial.com/ko/dot-net/topic/9059/-net-

3: .NET JSON Newtonsoft.Json

NuGet `Newtonsoft.Json` .NET JSON . .

Examples

JSON

```
using Newtonsoft.Json;

var obj = new Person
{
    Name = "Joe Smith",
    Age = 21
};
var serializedJson = JsonConvert.SerializeObject(obj);
```

JSON : `{"Name":"Joe Smith","Age":21}`

JSON .

```
var json = "{\"Name\":\"Joe Smith\",\"Age\":21}";
var person = JsonConvert.DeserializeObject<Person>(json);
```

"Joe Smith" 21 `Person` .

.NET JSON Newtonsoft.Json : https://riptutorial.com/ko/dot-net/topic/8746/-net-json-newtonsoft-json

4: ADO.NET

ADO (ActiveX Data Objects) .Net Microsoft SQL Server, Oracle XML . .Net ADO.Net , .

ADO.Net . .

`Parameters.AddWithValue` **SQL** **`Parameters.AddWithValue`** : `AddWithValue` . . . `char` / `varchar` ("n") `date` SQL Server .NET . `Add` .

Examples

SQL

```
// Uses Windows authentication. Replace the Trusted_Connection parameter with
// User Id=...;Password=...; to use SQL Server authentication instead. You may
// want to find the appropriate connection string for your server.
string connectionString =
@"Server=myServer\myInstance;Database=myDataBase;Trusted_Connection=True;"

string sql = "INSERT INTO myTable (myDateTimeField, myIntField) " +
    "VALUES (@someDateTime, @someInt);";

// Most ADO.NET objects are disposable and, thus, require the using keyword.
using (var connection = new SqlConnection(connectionString))
using (var command = new SqlCommand(sql, connection))
{
    // Use parameters instead of string concatenation to add user-supplied
    // values to avoid SQL injection and formatting issues. Explicitly supply datatype.

    // System.Data.SqlDbType is an enumeration. See Note1
    command.Parameters.Add("@someDateTime", SqlDbType.DateTime).Value = myDateTimeVariable;
    command.Parameters.Add("@someInt", SqlDbType.Int).Value = myInt32Variable;

    // Execute the SQL statement. Use ExecuteScalar and ExecuteReader instead
    // for query that return results (or see the more specific examples, once
    // those have been added).

    connection.Open();
    command.ExecuteNonQuery();
}
```

1 : MSFT SQL Server SqlDbType Enumeration .

2 : MySQL MySqlDbType .

- SQL

```
public void SaveNewEmployee(Employee newEmployee)
{
    // best practice - wrap all database connections in a using block so they are always
closed & disposed even in the event of an Exception
```

```
    // best practice - retrieve the connection string by name from the app.config or
web.config (depending on the application type) (note, this requires an assembly reference to
System.configuration)
    using(SqlConnection con = new
SqlConnection(System.Configuration.ConfigurationManager.ConnectionStrings["MyConnectionName"].Connectio

    {
        // best practice - use column names in your INSERT statement so you are not dependent
on the sql schema column order
        // best practice - always use parameters to avoid sql injection attacks and errors if
malformed text is used like including a single quote which is the sql equivalent of escaping
or starting a string (varchar/nvarchar)
        // best practice - give your parameters meaningful names just like you do variables in
your code
        using(SqlCommand sc = new SqlCommand("INSERT INTO employee (FirstName, LastName,
DateOfBirth /*etc*/) VALUES (@firstName, @lastName, @dateOfBirth /*etc*/)", con))
        {
            // best practice - always specify the database data type of the column you are
using
            // best practice - check for valid values in your code and/or use a database
constraint, if inserting NULL then use System.DbNull.Value
            sc.Parameters.Add(new SqlParameter("@firstName", SqlDbType.VarChar, 200){Value =
newEmployee.FirstName ?? (object) System.DBNull.Value});
            sc.Parameters.Add(new SqlParameter("@lastName", SqlDbType.VarChar, 200){Value =
newEmployee.LastName ?? (object) System.DBNull.Value});

            // best practice - always use the correct types when specifying your parameters,
Value is assigned to a DateTime instance and not a string representation of a Date
            sc.Parameters.Add(new SqlParameter("@dateOfBirth", SqlDbType.Date){ Value =
newEmployee.DateOfBirth });

            // best practice - open your connection as late as possible unless you need to
verify that the database connection is valid and wont fail and the proceeding code execution
takes a long time (not the case here)
            con.Open();
            sc.ExecuteNonQuery();
        }

        // the end of the using block will close and dispose the SqlConnection
        // best practice - end the using block as soon as possible to release the database
connection
    }
}

// supporting class used as parameter for example
public class Employee
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public DateTime DateOfBirth { get; set; }
}
```

ADO.NET

- . . = 100. SQL Server . SQL Server
- . (C #) .
- app.config web.config ( ).
    - System.configuration

.
    - .
- SQL
    - sql (varchar / nvarchar) .
    -
- SQL / / .
    - Sql .
    - .
    - (" , ").
    - (: DateTime DateTime).
    - . SQL Server . MAX -1 .
    - AddWithValue . / . AddWithValue ? .
- . .
    - (: `SqlConnection`). " ".
        - .
        - .

.

```
var providerName = "System.Data.SqlClient";    //Oracle.ManagedDataAccess.Client, IBM.Data.DB2
var connectionString = "{your-connection-string}";
//you will probably get the above two values in the ConnectionStringSettings object from
.config file

var factory = DbProviderFactories.GetFactory(providerName);
using(var connection = factory.CreateConnection()) {    //IDbConnection
    connection.ConnectionString = connectionString;
    connection.Open();

    using(var command = connection.CreateCommand()) {    //IDbCommand
        command.CommandText = "{query}";

        using(var reader = command.ExecuteReader()) {    //IDataReader
            while(reader.Read()) {
                ...
            }
        }
    }
}
```

ADO.NET : https://riptutorial.com/ko/dot-net/topic/3589/ado-net

5: ASP.NET ASP.NET MVC

ASP.NET

.resx . `HttpModule` .po . [. [delplex] . `HttpModule` . .

.po . .po .

Examples

1. MVC I18N .
2. web.config `<httpModules> <modules> i18n.LocalizingModule .`

```
<!-- IIS 6 -->
<httpModules>
  <add name="i18n.LocalizingModule" type="i18n.LocalizingModule, i18n" />
</httpModules>

<!-- IIS 7 -->
<system.webServer>
  <modules>
    <add name="i18n.LocalizingModule" type="i18n.LocalizingModule, i18n" />
  </modules>
</system.webServer>
```

3. "locale" . . : `/locale/fr/` .
4. `messages.po` .
5. `messages.po` .

```
#: Translation test
msgid "Hello, world!"
msgstr "Bonjour le monde!"
```

6. .

```
using System.Web.Mvc;

namespace I18nDemo.Controllers
{
    public class DefaultController : Controller
    {
        public ActionResult Index()
        {
            // Text inside [[[triple brackets]]] must precisely match
            // the msgid in your .po file.
            return Content("[[[Hello, world!]]]");
        }
    }
}
```

7. MVC (: http : // localhost : [yourportnumber] / default) .

URL (:
http : // localhost : [ ] / en / default .

8. `/en/` URL `/fr/` (.)
9. `/default` `/default` . URL .
10. web.config `locale` .

```
<!-- IIS 6 -->
<system.web>
  <httpHandlers>
    <add path="*" verb="*" type="System.Web.HttpNotFoundHandler"/>
  </httpHandlers>
</system.web>

<!-- IIS 7 -->
<system.webServer>
  <handlers>
    <remove name="BlockViewHandler"/>
   <add name="BlockViewHandler" path="*" verb="*" preCondition="integratedMode"
type="System.Web.HttpNotFoundHandler"/>
  </handlers>
</system.webServer>
```

ASP.NET ASP.NET MVC : https://riptutorial.com/ko/dot-net/topic/5086/asp-net---asp-net-mvc-

6: C # SHA1

SHA1 . SHA1 . : https://github.com/mahdiabasi/SHA1Tool

Examples

SHA1

System.Security.Cryptography System.IO .

```
public string GetSha1Hash(string filePath)
    {
        using (FileStream fs = File.OpenRead(filePath))
        {
            SHA1 sha = new SHA1Managed();
            return BitConverter.ToString(sha.ComputeHash(fs));
        }
    }
```

C # SHA1 : https://riptutorial.com/ko/dot-net/topic/9457/c-sharp-sha1-

7: C # SHA1

SHA1 . SHA1 .

github compelete : https://github.com/mahdiabasi/SHA1Tool

Examples

SHA1

System.Security.Cryptography .

```
public string GetSha1Hash(string filePath)
    {
        using (FileStream fs = File.OpenRead(filePath))
        {
            SHA1 sha = new SHA1Managed();
            return BitConverter.ToString(sha.ComputeHash(fs));
        }
    }
```

#

```
  public static string TextToHash(string text)
    {
        var sh = SHA1.Create();
        var hash = new StringBuilder();
        byte[] bytes = Encoding.UTF8.GetBytes(text);
        byte[] b = sh.ComputeHash(bytes);
        foreach (byte a in b)
        {
            var h = a.ToString("x2");
            hash.Append(h);
        }
        return hash.ToString();
    }
```

C # SHA1 : https://riptutorial.com/ko/dot-net/topic/9458/c-sharp-sha1-

8: CLR

Examples

CLR (Common Language Runtime) .NET Framework . :

- **Common Intermediate Language** (CIL IL)
- Just-In-Time
-
- AppDomains
-

CLR CLR ( ) . .

CLR : https://riptutorial.com/ko/dot-net/topic/3942/clr

9: DateTime

Examples

ParseExact

```
var dateString = "2015-11-24";

var date = DateTime.ParseExact(dateString, "yyyy-MM-dd", null);
Console.WriteLine(date);
```

2015 11 24 12:00:00 AM

CultureInfo.CurrentCulture null . .

.

```
var date = DateTime.ParseExact("24|201511", "dd|yyyyMM", null);
Console.WriteLine(date);
```

2015 11 24 12:00:00 AM

.

```
var date = DateTime.ParseExact("2015|11|24", "yyyy|MM|dd", null);
Console.WriteLine(date);
```

2015 11 24 12:00:00 AM

```
var date = DateTime.ParseExact("2015-01-24 11:11:30", "yyyy-mm-dd hh:MM:ss", null);
Console.WriteLine(date);
```

2011 11 24 11:01:30

.

.

```
var date = DateTime.ParseExact("11/24/2015", "d", new CultureInfo("en-US"));
var date = DateTime.ParseExact("2015-11-24T10:15:45", "s", null);
var date = DateTime.ParseExact("2015-11-24 10:15:45Z", "u", null);
```

ArgumentNullException

```
var date = DateTime.ParseExact(null, "yyyy-MM-dd", null);
var date = DateTime.ParseExact("2015-11-24", null, null);
```

FormatException

```
var date = DateTime.ParseExact("", "yyyy-MM-dd", null);
var date = DateTime.ParseExact("2015-11-24", "", null);
var date = DateTime.ParseExact("2015-0C-24", "yyyy-MM-dd", null);
var date = DateTime.ParseExact("2015-11-24", "yyyy-QQ-dd", null);

// Single-character format strings must be one of the standard formats
var date = DateTime.ParseExact("2015-11-24", "q", null);

// Format strings must match the input exactly* (see next section)
var date = DateTime.ParseExact("2015-11-24", "d", null); // Expects 11/24/2015 or 24/11/2015
for most cultures
```

```
var date = DateTime.ParseExact("2015-11-24T10:15:45",
  new [] { "s", "t", "u", "yyyy-MM-dd" }, // Will succeed as long as input matches one of
these
  CultureInfo.CurrentCulture, DateTimeStyles.None);
```

```
var dateString = "10/11/2015";
var date = DateTime.ParseExact(dateString, "d", new CultureInfo("en-US"));
Console.WriteLine("Day: {0}; Month: {1}", date.Day, date.Month);
```

Day : 11; : 10

```
date = DateTime.ParseExact(dateString, "d", new CultureInfo("en-GB"));
Console.WriteLine("Day: {0}; Month: {1}", date.Day, date.Month);
```

: 10; : 11

TryParse

DateTime . out .

out out DateTime.MinValue .

TryParse (string, out DateTime)

```
DateTime parsedValue;

if (DateTime.TryParse("monkey", out parsedValue))
{
   Console.WriteLine("Apparently, 'monkey' is a date/time value. Who knew?");
}
```

ISO 8601 .

```
DateTime.TryParse("11/24/2015 14:28:42", out parsedValue); // true
DateTime.TryParse("2015-11-24 14:28:42", out parsedValue); // true
DateTime.TryParse("2015-11-24T14:28:42", out parsedValue); // true
DateTime.TryParse("Sat, 24 Nov 2015 14:28:42", out parsedValue); // true
```

culture . .

```
// System set to en-US culture
bool result = DateTime.TryParse("24/11/2015", out parsedValue);
Console.WriteLine(result);
```

```
// System set to en-GB culture
bool result = DateTime.TryParse("11/24/2015", out parsedValue);
Console.WriteLine(result);
```

```
// System set to en-GB culture
bool result = DateTime.TryParse("10/11/2015", out parsedValue);
Console.WriteLine(result);
```

10 11 11 10 .

TryParse (, IFormatProvider, DateTimeStyles, out DateTime)

```
if (DateTime.TryParse(" monkey ", new CultureInfo("en-GB"),
    DateTimeStyles.AllowLeadingWhite | DateTimeStyles.AllowTrailingWhite, out parsedValue)
{
    Console.WriteLine("Apparently, ' monkey ' is a date/time value. Who knew?");
}
```

. `IFormatProvider` `null` culture .

throw . `IFormatProvider` `DateTimeStyles` .

- `NotSupportedException` : `IFormatProvider` .
- `ArgumentException` : `DateTimeStyles` `AssumeLocal` `AssumeUniversal` .

TryParseExact

`TryParse` `ParseExact` . throw Boolean .

TryParseExact (, , IFormatProvider, DateTimeStyles, out DateTime)

. .

```
DateTime.TryParseExact("11242015", "MMddyyyy", null, DateTimeStyles.None, out parsedValue); //
true
```

TryParseExact (, [], IFormatProvider, DateTimeStyles, out DateTime)

. .

```
DateTime.TryParseExact("11242015", new [] { "yyyy-MM-dd", "MMddyyyy" }, null,
DateTimeStyles.None, out parsedValue); // true
```

DateTime : https://riptutorial.com/ko/dot-net/topic/58/datetime--

10: HTTP

Examples

HTTP (HttpListener)

:

.

.

ASCII (*Content-Disposition*) .

```
using System;
using System.IO;
using System.Net;

class HttpFileServer
{
    private static HttpListenerResponse response;
    private static HttpListener listener;
    private static string baseFilesystemPath;

    static void Main(string[] args)
    {
        if (!HttpListener.IsSupported)
        {
            Console.WriteLine(
                "*** HttpListener requires at least Windows XP SP2 or Windows Server 2003.");
            return;
        }

        if(args.Length < 2)
        {
            Console.WriteLine("Basic read-only HTTP file server");
            Console.WriteLine();
            Console.WriteLine("Usage: httpfileserver <base filesystem path> <port>");
            Console.WriteLine("Request format: http://url:port/path/to/file.ext");
            return;
        }

        baseFilesystemPath = Path.GetFullPath(args[0]);
        var port = int.Parse(args[1]);

        listener = new HttpListener();
        listener.Prefixes.Add("http://*:" + port + "/");
        listener.Start();

        Console.WriteLine("--- Server stated, base path is: " + baseFilesystemPath);
        Console.WriteLine("--- Listening, exit with Ctrl-C");
        try
        {
            ServerLoop();
        }
```

```
        catch(Exception ex)
        {
            Console.WriteLine(ex);
            if(response != null)
            {
                SendErrorResponse(500, "Internal server error");
            }
        }
    }

    static void ServerLoop()
    {
        while(true)
        {
            var context = listener.GetContext();

            var request = context.Request;
            response = context.Response;
            var fileName = request.RawUrl.Substring(1);
            Console.WriteLine(
                "--- Got {0} request for: {1}",
                request.HttpMethod, fileName);

            if (request.HttpMethod.ToUpper() != "GET")
            {
                SendErrorResponse(405, "Method must be GET");
                continue;
            }

            var fullFilePath = Path.Combine(baseFilesystemPath, fileName);
            if(!File.Exists(fullFilePath))
            {
                SendErrorResponse(404, "File not found");
                continue;
            }

            Console.Write("    Sending file...");
            using (var fileStream = File.OpenRead(fullFilePath))
            {
                response.ContentType = "application/octet-stream";
                response.ContentLength64 = (new FileInfo(fullFilePath)).Length;
                response.AddHeader(
                    "Content-Disposition",
                    "Attachment; filename=\"" + Path.GetFileName(fullFilePath) + "\"");
                fileStream.CopyTo(response.OutputStream);
            }

            response.OutputStream.Close();
            response = null;
            Console.WriteLine(" Ok!");
        }
    }

    static void SendErrorResponse(int statusCode, string statusResponse)
    {
        response.ContentLength64 = 0;
        response.StatusCode = statusCode;
        response.StatusDescription = statusResponse;
        response.OutputStream.Close();
        Console.WriteLine("*** Sent error: {0} {1}", statusCode, statusResponse);
    }
```

```
}
```

HTTP (ASP.NET)

1 - .

2 - `project.json` (`rootDirectory`).

```
{
  "dependencies": {
    "Microsoft.AspNet.Server.Kestrel": "1.0.0-rc1-final",
    "Microsoft.AspNet.StaticFiles": "1.0.0-rc1-final"
  },

  "commands": {
    "web": "Microsoft.AspNet.Server.Kestrel --server.urls http://localhost:60000"
  },

  "frameworks": {
    "dnxcore50": { }
  },

  "fileServer": {
    "rootDirectory": "c:\\users\\username\\Documents"
  }
}
```

3 - `Startup.cs` .

```
using System;
using Microsoft.AspNet.Builder;
using Microsoft.AspNet.FileProviders;
using Microsoft.AspNet.Hosting;
using Microsoft.AspNet.StaticFiles;
using Microsoft.Extensions.Configuration;

public class Startup
{
    public void Configure(IApplicationBuilder app)
    {
        var builder = new ConfigurationBuilder();
        builder.AddJsonFile("project.json");
        var config = builder.Build();
        var rootDirectory = config["fileServer:rootDirectory"];
        Console.WriteLine("File server root directory: " + rootDirectory);

        var fileProvider = new PhysicalFileProvider(rootDirectory);

        var options = new StaticFileOptions();
        options.ServeUnknownFileTypes = true;
        options.FileProvider = fileProvider;
        options.OnPrepareResponse = context =>
        {
            context.Context.Response.ContentType = "application/octet-stream";
            context.Context.Response.Headers.Add(
                "Content-Disposition",
                $"Attachment; filename=\"{context.File.Name}\"");
```

```
        };

        app.UseStaticFiles(options);
    }
}
```

4 - .

```
dnvm use 1.0.0-rc1-final -r coreclr -p
dnu restore
```

: . `dnvm list` CLR .

5 - `dnx web` . `http://localhost:60000/path/to/file.ext` .

ASCII (Content-Disposition)

HTTP : https://riptutorial.com/ko/dot-net/topic/53/http-

11: HTTP

HTTP / 1.1 RFC .

- 7230 :
- 7231 :
- 7232 :
- 7233 :
- 7234 :
- 7235 :
- 7239 : HTTP
- 7240 : HTTP

RFC .

- 7236 :
- 7237 :

RFC :

- 7238 : 308 ( )

:

- 4918 : WebDAV (Web Distributed Authoring and Versioning) HTTP
- 4791 : WebDAV (CalDAV)

Examples

System.Net.HttpWebRequest GET

```
string requestUri = "http://www.example.com";
string responseData;

HttpWebRequest request = (HttpWebRequest)WebRequest.Create(parameters.Uri);
WebResponse response = request.GetResponse();

using (StreamReader responseReader = new StreamReader(response.GetResponseStream()))
{
    responseData = responseReader.ReadToEnd();
}
```

System.Net.WebClient GET

```
string requestUri = "http://www.example.com";
string responseData;

using (var client = new WebClient())
{
```

```
    responseData = client.DownloadString(requestUri);
}
```

System.Net.HttpClient GET

`HttpClient` NuGet : Microsoft HTTP .

```
string requestUri = "http://www.example.com";
string responseData;

using (var client = new HttpClient())
{
    using(var response = client.GetAsync(requestUri).Result)
    {
       response.EnsureSuccessStatusCode();
       responseData = response.Content.ReadAsStringAsync().Result;
    }
}
```

System.Net.HttpWebRequest POST

```
string requestUri = "http://www.example.com";
string requestBodyString = "Request body string.";
string contentType = "text/plain";
string requestMethod = "POST";

HttpWebRequest request = (HttpWebRequest)WebRequest.Create(requestUri)
{
  Method = requestMethod,
  ContentType = contentType,
};

byte[] bytes = Encoding.UTF8.GetBytes(requestBodyString);
Stream stream = request.GetRequestStream();
stream.Write(bytes, 0, bytes.Length);
stream.Close();

HttpWebResponse response = (HttpWebResponse)request.GetResponse();
```

System.Net.WebClient POST

```
string requestUri = "http://www.example.com";
string requestBodyString = "Request body string.";
string contentType = "text/plain";
string requestMethod = "POST";

byte[] responseBody;
byte[] requestBodyBytes = Encoding.UTF8.GetBytes(requestBodyString);

using (var client = new WebClient())
{
    client.Headers[HttpRequestHeader.ContentType] = contentType;
    responseBody = client.UploadData(requestUri, requestMethod, requestBodyBytes);
}
```

System.Net.HttpClient POST

`HttpClient` NuGet : Microsoft HTTP .

```
string requestUri = "http://www.example.com";
string requestBodyString = "Request body string.";
string contentType = "text/plain";
string requestMethod = "POST";

var request = new HttpRequestMessage
{
    RequestUri = requestUri,
    Method = requestMethod,
};

byte[] requestBodyBytes = Encoding.UTF8.GetBytes(requestBodyString);
request.Content = new ByteArrayContent(requestBodyBytes);

request.Content.Headers.ContentType = new MediaTypeHeaderValue(contentType);

HttpResponseMessage result = client.SendAsync(request).Result;
result.EnsureSuccessStatusCode();
```

System.Net.Http.HttpClient HTTP

```
using System;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Threading.Tasks;

class HttpGet
{
    private static async Task DownloadAsync(string fromUrl, string toFile)
    {
        using (var fileStream = File.OpenWrite(toFile))
        {
            using (var httpClient = new HttpClient())
            {
                Console.WriteLine("Connecting...");
                using (var networkStream = await httpClient.GetStreamAsync(fromUrl))
                {
                    Console.WriteLine("Downloading...");
                    await networkStream.CopyToAsync(fileStream);
                    await fileStream.FlushAsync();
                }
            }
        }
    }

    static void Main(string[] args)
    {
        try
        {
            Run(args).Wait();
        }
        catch (Exception ex)
        {
```

```
            if (ex is AggregateException)
                ex = ((AggregateException)ex).Flatten().InnerExceptions.First();

            Console.WriteLine("--- Error: " +
                (ex.InnerException?.Message ?? ex.Message));
        }
    }
    static async Task Run(string[] args)
    {
        if (args.Length < 2)
        {
            Console.WriteLine("Basic HTTP downloader");
            Console.WriteLine();
            Console.WriteLine("Usage: httpget <url>[<:port>] <file>");
            return;
        }

        await DownloadAsync(fromUrl: args[0], toFile: args[1]);

        Console.WriteLine("Done!");
    }
}
```

HTTP : https://riptutorial.com/ko/dot-net/topic/32/http-

12: JIT

JIT JIT (just-in-time) (compilation) . JIT .NET . CLR (C #, F #, Visual Basic) Interpreted Language IL . . .

JIT ?

- : CLR IL IL .
- : JIT ( )

JIT Wikipedia : https://en.wikipedia.org/wiki/Just-in-time_compilation

Examples

Hello World :

```
using System;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World");
        }
    }
}
```

(JIT)

```
//  Microsoft (R) .NET Framework IL Disassembler.  Version 4.6.1055.0
//  Copyright (c) Microsoft Corporation.  All rights reserved.



// Metadata version: v4.0.30319
.assembly extern mscorlib
{
  .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )                         // .z\V.4..
  .ver 4:0:0:0
}
.assembly HelloWorld
{
  .custom instance void
[mscorlib]System.Runtime.CompilerServices.CompilationRelaxationsAttribute::.ctor(int32) = ( 01
00 08 00 00 00 00 00 )
  .custom instance void
[mscorlib]System.Runtime.CompilerServices.RuntimeCompatibilityAttribute::.ctor() = ( 01 00 01
00 54 02 16 57 72 61 70 4E 6F 6E 45 78   // ....T..WrapNonEx

63 65 70 74 69 6F 6E 54 68 72 6F 77 73 01 )       // ceptionThrows.
```

```
  // --- The following custom attribute is added automatically, do not uncomment -------
  //  .custom instance void [mscorlib]System.Diagnostics.DebuggableAttribute::.ctor(valuetype
[mscorlib]System.Diagnostics.DebuggableAttribute/DebuggingModes) = ( 01 00 07 01 00 00 00 00 )


  .custom instance void [mscorlib]System.Reflection.AssemblyTitleAttribute::.ctor(string) = (
01 00 0A 48 65 6C 6C 6F 57 6F 72 6C 64 00 00 )    // ...HelloWorld..
  .custom instance void
[mscorlib]System.Reflection.AssemblyDescriptionAttribute::.ctor(string) = ( 01 00 00 00 00 )
  .custom instance void
[mscorlib]System.Reflection.AssemblyConfigurationAttribute::.ctor(string) = ( 01 00 00 00 00 )

  .custom instance void [mscorlib]System.Reflection.AssemblyCompanyAttribute::.ctor(string) =
( 01 00 00 00 00 )
  .custom instance void [mscorlib]System.Reflection.AssemblyProductAttribute::.ctor(string) =
( 01 00 0A 48 65 6C 6C 6F 57 6F 72 6C 64 00 00 )    // ...HelloWorld..
  .custom instance void [mscorlib]System.Reflection.AssemblyCopyrightAttribute::.ctor(string)
= ( 01 00 12 43 6F 70 79 72 69 67 68 74 20 C2 A9 20   // ...Copyright ..

20 32 30 31 37 00 00 )                            //  2017..
  .custom instance void [mscorlib]System.Reflection.AssemblyTrademarkAttribute::.ctor(string)
= ( 01 00 00 00 00 )
  .custom instance void
[mscorlib]System.Runtime.InteropServices.ComVisibleAttribute::.ctor(bool) = ( 01 00 00 00 00 )

  .custom instance void [mscorlib]System.Runtime.InteropServices.GuidAttribute::.ctor(string)
= ( 01 00 24 33 30 38 62 33 64 38 36 2D 34 31 37 32   // ..$308b3d86-4172

2D 34 30 32 32 2D 61 66 63 63 2D 33 66 38 65 33   // -4022-afcc-3f8e3

32 33 33 63 35 62 30 00 00 )                      // 233c5b0..
  .custom instance void
[mscorlib]System.Reflection.AssemblyFileVersionAttribute::.ctor(string) = ( 01 00 07 31 2E 30
2E 30 2E 30 00 00 )             // ...1.0.0.0..
  .custom instance void
[mscorlib]System.Runtime.Versioning.TargetFrameworkAttribute::.ctor(string) = ( 01 00 1C 2E 4E
45 54 46 72 61 6D 65 77 6F 72 6B   // ....NETFramework

2C 56 65 72 73 69 6F 6E 3D 76 34 2E 35 2E 32 01   // ,Version=v4.5.2.

00 54 0E 14 46 72 61 6D 65 77 6F 72 6B 44 69 73   // .T..FrameworkDis

70 6C 61 79 4E 61 6D 65 14 2E 4E 45 54 20 46 72   // playName..NET Fr

61 6D 65 77 6F 72 6B 20 34 2E 35 2E 32 )          // amework 4.5.2
  .hash algorithm 0x00008004
  .ver 1:0:0:0
}
.module HelloWorld.exe
// MVID: {2A7E1D59-1272-4B47-85F6-D7E1ED057831}
.imagebase 0x00400000
.file alignment 0x00000200
.stackreserve 0x00100000
.subsystem 0x0003       // WINDOWS_CUI
.corflags 0x00020003    //  ILONLY 32BITPREFERRED
// Image base: 0x0000021C70230000


// =============== CLASS MEMBERS DECLARATION ===================

.class private auto ansi beforefieldinit HelloWorld.Program
```

```
       extends [mscorlib]System.Object
{
  .method private hidebysig static void  Main(string[] args) cil managed
  {
    .entrypoint
    // Code size       13 (0xd)
    .maxstack  8
    IL_0000:  nop
    IL_0001:  ldstr      "Hello World"
    IL_0006:  call       void [mscorlib]System.Console::WriteLine(string)
    IL_000b:  nop
    IL_000c:  ret
  } // end of method Program::Main

  .method public hidebysig specialname rtspecialname
          instance void  .ctor() cil managed
  {
    // Code size       8 (0x8)
    .maxstack  8
    IL_0000:  ldarg.0
    IL_0001:  call       instance void [mscorlib]System.Object::.ctor()
    IL_0006:  nop
    IL_0007:  ret
  } // end of method Program::.ctor

} // end of class HelloWorld.Program
```

MS ILDASM (IL)

JIT : https://riptutorial.com/ko/dot-net/topic/9222/jit-

13: JSON

JavaScriptSerializer Json.NET

`JavaScriptSerializer` .NET 3.5 AJAX .NET . JSON .

`JavaScriptSerializer` Json.NET . Json.NET JSON (`JavaScriptConverter` `JavaScriptSerializer`).

Examples

System.Web.Script.Serialization.JavaScriptSerializer

`JavaScriptSerializer.Deserialize<T>(input)` `JavaScriptSerializer` JSON `<T>` .

```
using System.Collections;
using System.Web.Script.Serialization;

// ...

string rawJSON = "{\"Name\":\"Fibonacci Sequence\",\"Numbers\":[0, 1, 1, 2, 3, 5, 8, 13]}";

JavaScriptSerializer JSS = new JavaScriptSerializer();
Dictionary<string, object> parsedObj = JSS.Deserialize<Dictionary<string, object>>(rawJSON);

string name = parsedObj["Name"].toString();
ArrayList numbers = (ArrayList)parsedObj["Numbers"]
```

: `JavaScriptSerializer` .NET 3.5 .

Json.NET

```
internal class Sequence{
    public string Name;
    public List<int> Numbers;
}

// ...

string rawJSON = "{\"Name\":\"Fibonacci Sequence\",\"Numbers\":[0, 1, 1, 2, 3, 5, 8, 13]}";

Sequence sequence = JsonConvert.DeserializeObject<Sequence>(rawJSON);
```

Json.NET .

: Json.NET .NET 2 .

Json.NET

```
[JsonObject("person")]
```

```
public class Person
{
    [JsonProperty("name")]
    public string PersonName { get; set; }
    [JsonProperty("age")]
    public int PersonAge { get; set; }
    [JsonIgnore]
    public string Address { get; set; }
}

Person person = new Person { PersonName = "Andrius", PersonAge = 99, Address = "Some address"
};
string rawJson = JsonConvert.SerializeObject(person);

Console.WriteLine(rawJson); // {"name":"Andrius","age":99}
```

( ) json json (JsonIgnore) .

Json.NET .

C # *PascalCase* . JSON *camelCase* . .

```
using Newtonsoft.Json;
using Newtonsoft.Json.Serialization;

public class Person
{
    public string Name { get; set; }
    public int Age { get; set; }
    [JsonIgnore]
    public string Address { get; set; }
}

public void ToJson() {
    Person person = new Person { Name = "Andrius", Age = 99, Address = "Some address" };
    var resolver = new CamelCasePropertyNamesContractResolver();
    var settings = new JsonSerializerSettings { ContractResolver = resolver };
    string json = JsonConvert.SerializeObject(person, settings);

    Console.WriteLine(json); // {"name":"Andrius","age":99}
}
```

- Newtonsoft.Json

.

```
using Newtonsoft.Json;

var rawJSON      = "{\"Name\":\"Fibonacci Sequence\",\"Numbers\":[0, 1, 1, 2, 3, 5, 8, 13]}";
var fibo         = JsonConvert.DeserializeObject<Dictionary<string, object>>(rawJSON);
var rawJSON2     = JsonConvert.SerializeObject(fibo);
```

Newtonsoft Json.NET (ExpandoObject /) json .

```
dynamic jsonObject = new ExpandoObject();
jsonObject.Title   = "Merchent of Venice";
```

```
jsonObject.Author  = "William Shakespeare";
Console.WriteLine(JsonConvert.SerializeObject(jsonObject));
```

```
var rawJson = "{\"Name\":\"Fibonacci Sequence\",\"Numbers\":[0, 1, 1, 2, 3, 5, 8, 13]}";
dynamic parsedJson = JObject.Parse(rawJson);
Console.WriteLine("Name: " + parsedJson.Name);
Console.WriteLine("Name: " + parsedJson.Numbers.Length);
```

rawJson .

JSON / . / Json .

JsonSerializerSettings Json.NET

serializer Null .net json serializer . `JsonSerializerSettings` .

```
public static string Serialize(T obj)
{
   string result = JsonConvert.SerializeObject(obj, new JsonSerializerSettings {
NullValueHandling = NullValueHandling.Ignore});
   return result;
}
```

.net . `List<Student>` . `JsonSerializerSettings` :

```
public static string Serialize(T obj)
{
   string result = JsonConvert.SerializeObject(obj, new JsonSerializerSettings {
ReferenceLoopHandling = ReferenceLoopHandling.Ignore});
   return result;
}
```

.

```
public static string Serialize(T obj)
{
   string result = JsonConvert.SerializeObject(obj, new JsonSerializerSettings {
NullValueHandling = NullValueHandling.Ignore, ReferenceLoopHandling =
ReferenceLoopHandling.Ignore});
   return result;
}
```

JSON : https://riptutorial.com/ko/dot-net/topic/183/json-

14: LINQ

LINQ (Language Integrated Query) . LINQ . LINQ . XML , SQL , ADO.NET , .NET . LINQ C # VB .

- public static TSource Aggregate <TSource> (IEnumerable <TSource> , Func <TSource, TSource, TSource> func)
- public static TAccumulate Aggregate <TSource, TAccumulate> (IEnumerable <TSource> , TAccumulate , Func <TAccumulate, TSource, TAccumulate>)
- public static TResult Aggregate <TSource, TAccumulate, TResult> (IEnumerable <TSource> , TAccumulate , Func <TAccumulate, TSource, TAccumulate> , Func <TAccumulate, TResult> resultSelector)
- public static Boolean <TSource> (IEnumerable <TSource> , Func <TSource, >)
- public static Boolean Any <TSource> (IEnumerable <TSource>)
- public static Boolean <TSource> (IEnumerable <TSource> , Func <TSource, >)
- public static IEnumerable <TSource> AsEnumerable <TSource> (IEnumerable <TSource>)
- public static Decimal Average (IEnumerable <Decimal>)
- public static Double Average (IEnumerable <Double>)
- public static Double Average (IEnumerable <Int32>)
- public static Double Average (IEnumerable <Int64>)
- public static Nullable <Decimal> Average (IEnumerable <Nullable <Decimal >>)
- public static Nullable <Double> Average (IEnumerable <Nullable <Double >>)
- public static Nullable <Double> Average (IEnumerable <Nullable <Int32 >>)
- public static Nullable <Double> Average (IEnumerable <Nullable <Int64 >>)
- public static Nullable <Single> Average (IEnumerable <Nullable <Single >>)
- public static Single Average (IEnumerable <Single>)
- public static Decimal Average <TSource> (IEnumerable <TSource> , Func <TSource, Decimal>)
- public static Double Average <TSource> (IEnumerable <TSource> , Func <TSource, Double>)
- Double <TSource> (IEnumerable <TSource> , Func <TSource, Int32>)
- public static Double Average <TSource> (IEnumerable <TSource> , Func <TSource, Int64>)
- public static Nullable <Decimal> Average <TSource> (IEnumerable <TSource> , Func <TSource, Nullable <Decimal >> selector)
- public static Nullable <Double> Average <TSource> (IEnumerable <TSource> , Func <TSource, Nullable <Double >> selector)
- public static Nullable <Double> Average <TSource> (IEnumerable <TSource> , Func <TSource, Nullable <Int32 >> selector)
- public static Nullable <Double> <TSource> (IEnumerable <TSource> , Func <TSource, Nullable <Int64 >> selector)
- public static Nullable <Single> Average <TSource> (IEnumerable <TSource> , Func <TSource, Nullable <Single >> selector)
- public static Single Average <TSource> (IEnumerable <TSource> , Func <TSource, Single> selector)

- public static IEnumerable <TResult> <TResult> (IEnumerable)
- public static IEnumerable <TSource> Concat <TSource> (IEnumerable <TSource> , IEnumerable <TSource>)
- public static Boolean <TSource> (IEnumerable <TSource> , TSource)
- public static Boolean <TSource> (IEnumerable <TSource> , TSource , IEqualityComparer <TSource>)
- public static Int32 Count <TSource> (IEnumerable <TSource>)
- public static Int32 Count <TSource> (IEnumerable <TSource> , Func <TSource, >)
- public static IEnumerable <TSource> DefaultIfEmpty <TSource> (IEnumerable <TSource>)
- public static IEnumerable <TSource> DefaultIfEmpty <TSource> (IEnumerable <TSource> , TSource defaultValue)
- public static IEnumerable <TSource>   <TSource> (IEnumerable <TSource>)
- public static IEnumerable <TSource>   <TSource> (IEnumerable <TSource> , IEqualityComparer <TSource>)
- public static TSource ElementAt <TSource> (IEnumerable <TSource> , Int32)
- public static TSource ElementAtOrDefault <TSource> (IEnumerable <TSource> , Int32)
- IEnumerable <TResult> <TResult> ()
- public static IEnumerable <TSource> <TSource> (IEnumerable <TSource> , IEnumerable <TSource> second)
- public static IEnumerable <TSource> <TSource> (IEnumerable <TSource> , IEnumerable <TSource> , IEqualityComparer <TSource> comparer)
- public static TSource   <TSource> (IEnumerable <TSource>)
- public static TSource   <TSource> (IEnumerable <TSource> , Func <TSource, >)
- public static TSource FirstOrDefault <TSource> (IEnumerable <TSource>)
- public static TSource FirstOrDefault <TSource> (IEnumerable <TSource> , Func <TSource, Boolean>)
- public static IEnumerable <IGKeying <TKey, TSource >> GroupBy <TSource, TKey> (IEnumerable <TSource> , Func <TSource, TKey> keySelector)
- public static IEnumerable <IGKeying <TKey, TSource >> GroupBy <TSource, TKey> (IEnumerable <TSource> , Func <TSource, TKey> keySelector, IEqualityComparer <TKey>)
- public IEnumerable <IGrouping <TKey, TElement >> GroupBy <TSource, TKey, TElement> (IEnumerable <TSource> , Func <TSource, TKey> keySelector, Func <TSource, TElement> elementSelector)
- IEnumerable <TSource> , Func <TSource, TKey> keySelector, Func <TSource, TElement> elementSelector, IEqualityComparer <TKey> comparer) IEnumerable <IGrouping <TKey, TElement >> GroupBy <TSource, TKey, TElement>
- public IEnumerable <TResult> GroupBy <TSource, TKey, TResult> (IEnumerable <TSource> , Func <TSource, TKey> , Func <TKey, IEnumerable <TSource>, TResult> resultSelector)
- IEnumerable <TSource> , Func <TSource, TKey> keySelector, Func <TKey, IEnumerable <TSource>, TResult> resultSelector, IEqualityComparer <TKey> comparer) IEnumerable <TResult> GroupBy <TSource, TKey, TResult>
- public static IEnumerable <TResult> GroupBy <TSource, TKey, TElement, TResult> (IEnumerable <TSource> , Func <TSource, TKey> , Func <TSource, TElement> , Func <TKey, IEnumerable <TElement>, TResult > resultSelector)
- public static IEnumerable <TResult> GroupBy <TSource, TKey, TElement, TResult> (

IEnumerable <TSource> , Func <TSource, TKey> , Func <TSource, TElement> , Func <TKey, IEnumerable <TElement>, TResult > resultSelector, IEqualityComparer <TKey>)

- public static IEnumerable <TResult> GroupJoin <TOuter, TInner, TKey, TResult> (IEnumerable <TOuter> , IEnumerable < IEnumerable, , Func <TOuter, TKey> outerKeySelector, Func <TInner, TKey> , Func <TOuter, IEnumerable <TInner>, TResult> resultSelector)
- public static IEnumerable <TResult> GroupJoin <TOuter, TInner, TKey, TResult> (IEnumerable <TOuter> , IEnumerable < IEnumerable, , Func <TOuter, TKey> outerKeySelector, Func <TInner, TKey> , Func <TOuter, IEnumerable <TInner>, TResult> resultSelector, IEqualityComparer <TKey> comparer)
- public static IEnumerable <TSource> Intersect <TSource> (IEnumerable <TSource> , IEnumerable <TSource>)
- public static IEnumerable <TSource> Intersect <TSource> (IEnumerable <TSource> first, IEnumerable <TSource> second, IEqualityComparer <TSource> comparer)
- public static IEnumerable <TResult> innerKeySelector, Func <TOuter, TKey, TResult> (IEnumerable <TOuter> , IEnumerable <TInner> , Func <TOuter, TKey> outerKeySelector, Func <TInner, TKey> TInner, TResult> resultSelector)
- public static IEnumerable <TResult> innerKeySelector, Func <TOuter, TKey, TResult> (IEnumerable <TOuter> , IEnumerable <TInner> , Func <TOuter, TKey> outerKeySelector, Func <TInner, TKey> TInner, TResult> resultSelector, IEqualityComparer <TKey>)
- public static TSource <TSource> (IEnumerable <TSource>)
- public static TSource Last <TSource> (IEnumerable <TSource> , Func <TSource, Boolean>)
- public static TSource LastOrDefault <TSource> (IEnumerable <TSource>)
- public static TSource LastOrDefault <TSource> (IEnumerable <TSource> , Func <TSource, Boolean>)
- public static Int64 LongCount <TSource> (IEnumerable <TSource>)
- public static Int64 LongCount <TSource> (IEnumerable <TSource> , Func <TSource, Boolean>)
- public static Decimal Max (IEnumerable <Decimal>)
- public static Double Max (IEnumerable <Double>)
- public static Int32 Max (IEnumerable <Int32>)
- public static Int64 Max (IEnumerable <Int64>)
- public static Nullable <Decimal> Max (IEnumerable <Nullable <Decimal >>)
- public static Nullable <Double> Max (IEnumerable <Nullable <Double >>)
- public static Nullable <Int32> Max (IEnumerable <Nullable <Int32 >>)
- public static Nullable <Int64> Max (IEnumerable <Nullable <Int64 >>)
- public static Nullable <Single> Max (IEnumerable <Nullable <Single >>)
- public static Single Max (IEnumerable <Single>)
- public static TSource Max <TSource> (IEnumerable <TSource>)
- public static Decimal Max <TSource> (IEnumerable <TSource> , Func <TSource, Decimal> selector)
- public static Double Max <TSource> (IEnumerable <TSource> , Func <TSource, Double>)
- public static Int32 Max <TSource> (IEnumerable <TSource> , Func <TSource, Int32>)
- public static Int64 Max <TSource> (IEnumerable <TSource> , Func <TSource, Int64>)
- public static Nullable <Decimal> Max <TSource> (IEnumerable <TSource> , Func

<TSource, Nullable <Decimal >> selector)
- public static Nullable <Double> Max <TSource> (IEnumerable <TSource> , Func <TSource, Nullable <Double >> selector)
- public static Nullable <Int32> Max <TSource> (IEnumerable <TSource> , Func <TSource, Nullable <Int32 >> selector)
- public static Nullable <Int64> Max <TSource> (IEnumerable <TSource> , Func <TSource, Nullable <Int64 >> selector)
- public static Nullable <Single> Max <TSource> (IEnumerable <TSource> , Func <TSource, Nullable <Single >> selector)
- public static Single Max <TSource> (IEnumerable <TSource> , Func <TSource, Single> selector)
- public static TResult Max <TSource, TResult> (IEnumerable <TSource> , Func <TSource, TResult> selector)
- Decimal Min (IEnumerable <Decimal>)
- public static Double Min (IEnumerable <Double>)
- public static Int32 Min (IEnumerable <Int32>)
- public static Int64 Min (IEnumerable <Int64>)
- public static Nullable <Decimal> Min (IEnumerable <Nullable <Decimal >>)
- public static Nullable <Double> Min (IEnumerable <Nullable <Double >>)
- public static Nullable <Int32> Min (IEnumerable <Nullable <Int32 >>)
- public static Nullable <Int64> Min (IEnumerable <Nullable <Int64 >>)
- public static Nullable <Single> Min (IEnumerable <Nullable <Single >>)
- public static Single Min (IEnumerable <Single>)
- public static TSource Min <TSource> (IEnumerable <TSource>)
- public static Decimal Min <TSource> (IEnumerable <TSource> , Func <TSource, Decimal>)
- Double Min <TSource> (IEnumerable <TSource> , Func <TSource, Double>)
- public static Int32 Min <TSource> (IEnumerable <TSource> , Func <TSource, Int32>)
- public static Int64 Min <TSource> (IEnumerable <TSource> , Func <TSource, Int64>)
- public static Nullable <Decimal> Min <TSource> (IEnumerable <TSource> , Func <TSource, Nullable <Decimal >> selector)
- public static Nullable <Double> Min <TSource> (IEnumerable <TSource> , Func <TSource, Nullable <Double >> selector)
- public static Nullable <Int32> Min <TSource> (IEnumerable <TSource> , Func <TSource, Nullable <Int32 >> selector)
- public static Nullable <Int64> Min <TSource> (IEnumerable <TSource> , Func <TSource, Nullable <Int64 >> selector)
- public static Nullable <Single> Min <TSource> (IEnumerable <TSource> , Func <TSource, Nullable <Single >> selector)
- public static Single Min <TSource> (IEnumerable <TSource> , Func <TSource, Single> selector)
- public static TResult Min <TSource, TResult> (IEnumerable <TSource> , Func <TSource, TResult> selector)
- public static IEnumerable <TResult> OfType <TResult> (IEnumerable)
- public static IOrderedEnumerable <TSource> OrderBy <TSource, TKey> (IEnumerable <TSource> , Func <TSource, TKey> keySelector)
- public static IOrderedEnumerable <TSource> OrderBy <TSource, TKey> (IEnumerable

<TSource> , Func <TSource, TKey> keySelector, IComparer <TKey>)

- public static IOrderedEnumerable <TSource> OrderByDescending <TSource, TKey> (IEnumerable <TSource> , Func <TSource, TKey> keySelector)
- public static IOrderedEnumerable <TSource> OrderByDescending <TSource, TKey> (IEnumerable <TSource> , Func <TSource, TKey> keySelector, IComparer <TKey>)
- IEnumerable <Int32> (Int32 , Int32)
- public static IEnumerable <TResult> <TResult> (TResult , Int32)
- public static IEnumerable <TSource> Reverse <TSource> (IEnumerable <TSource>)
- public static IEnumerable <TResult> <TSource, TResult> (IEnumerable <TSource> , Func <TSource, TResult>) .
- public static IEnumerable <TResult> <TSource, TResult> (IEnumerable <TSource> , Func <TSource, Int32, TResult> selector) .
- public static IEnumerable <TResult> SelectMany <TSource, TResult> (IEnumerable <TSource> , Func <TSource, IEnumerable <TResult >> selector)
- public static IEnumerable <TResult> SelectMany <TSource, TResult> (IEnumerable <TSource> , Func <TSource, Int32, IEnumerable <TResult >> selector)
- public static IEnumerable <TResult> SelectMany <TSource, TCollection, TResult> (IEnumerable <TSource> , Func <TSource, IEnumerable <TCollection >> , Func <TSource, TCollection, TResult> resultSelector)
- public static IEnumerable <TResult> SelectMany <TSource, TCollection, TResult> (IEnumerable <TSource> , Func <TSource, Int32, IEnumerable <TCollection >> , Func <TSource, TCollection, TResult> resultSelector)
- public static Boolean SequenceEqual <TSource> (IEnumerable <TSource> first, IEnumerable <TSource> second)
- public static Boolean SequenceEqual <TSource> (IEnumerable <TSource> , IEnumerable <TSource> , IEqualityComparer <TSource>)
- public static TSource Single <TSource> (IEnumerable <TSource>)
- public static TSource <TSource> (IEnumerable <TSource> , Func <TSource, >)
- public static TSource SingleOrDefault <TSource> (IEnumerable <TSource>)
- public static TSource SingleOrDefault <TSource> (IEnumerable <TSource> , Func <TSource, Boolean>)
- public static IEnumerable <TSource> Skip <TSource> (IEnumerable <TSource> , Int32)
- public static IEnumerable <TSource> SkipWhile <TSource> (IEnumerable <TSource> , Func <TSource, Boolean>)
- public static IEnumerable <TSource> SkipWhile <TSource> (IEnumerable <TSource> , Func <TSource, Int32, Boolean>)
- public static Decimal Sum (IEnumerable <Decimal>)
- public static Double Sum (IEnumerable <Double>)
- Int32 Sum (IEnumerable <Int32>)
- public static Int64 Sum (IEnumerable <Int64>)
- public static Nullable <Decimal> Sum (IEnumerable <Nullable <Decimal >>)
- public static Nullable <Double> Sum (IEnumerable <Nullable <Double >>)
- public static Nullable <Int32> Sum (IEnumerable <Nullable <Int32 >>)
- public static Nullable <Int64> Sum (IEnumerable <Nullable <Int64 >>)
- public static Nullable <Single> Sum (IEnumerable <Nullable <Single >>)
- public static Single Sum (IEnumerable <Single>)
-

Decimal Sum <TSource> (IEnumerable <TSource> , Func <TSource, Decimal> selector)

- public static Double Sum <TSource> (IEnumerable <TSource> , Func <TSource, Double>)
- static Int32 Sum <TSource> (IEnumerable <TSource> , Func <TSource, Int32> selector)
- public static Int64 Sum <TSource> (IEnumerable <TSource> , Func <TSource, Int64>)
- public static Nullable <Decimal> Sum <TSource> (IEnumerable <TSource> , Func <TSource, Nullable <Decimal >> selector)
- public static Nullable <Double> Sum <TSource> (IEnumerable <TSource> , Func <TSource, Nullable <Double >> selector)
- public static Nullable <Int32> Sum <TSource> (IEnumerable <TSource> , Func <TSource, Nullable <Int32 >> selector)
- public static Nullable <Int64> Sum <TSource> (IEnumerable <TSource> , Func <TSource, Nullable <Int64 >> selector)
- public static Nullable <Single> Sum <TSource> (IEnumerable <TSource> , Func <TSource, Nullable <Single >> selector)
- public static Single Sum <TSource> (IEnumerable <TSource> , Func <TSource, Single> selector)
- public static IEnumerable <TSource> Take <TSource> (IEnumerable <TSource> , Int32)
- public static IEnumerable <TSource> TakeWhile <TSource> (IEnumerable <TSource> , Func <TSource, Boolean>)
- public static IEnumerable <TSource> TakeWhile <TSource> (IEnumerable <TSource> , Func <TSource, Int32, Boolean>)
- public static IOrderedEnumerable <TSource> ThenBy <TSource, TKey> (IOrderedEnumerable <TSource> , Func <TSource, TKey> keySelector)
- public static IOrderedEnumerable <TSource> ThenBy <TSource, TKey> (IOrderedEnumerable <TSource> , Func <TSource, TKey> keySelector, IComparer <TKey>)
- public static IOrderedEnumerable <TSource> ThenByDescending <TSource, TKey> (IOrderedEnumerable <TSource> , Func <TSource, TKey> keySelector)
- public static IOrderedEnumerable <TSource> ThenByDescending <TSource, TKey> (IOrderedEnumerable <TSource> , Func <TSource, TKey> keySelector, IComparer <TKey>)
- public static TSource [] ToArray <TSource> (IEnumerable <TSource>)
- public static Dictionary <TKey, TSource> ToDictionary <TSource, TKey> (IEnumerable <TSource> , Func <TSource, TKey> keySelector)
- public static Dictionary <TKey, TSource> ToDictionary <TSource, TKey> (IEnumerable <TSource> , Func <TSource, TKey> keySelector, IEqualityComparer <TKey>)
- public static Dictionary <TKey, TElement> ToDictionary <TSource, TKey, TElement> (IEnumerable <TSource> , Func <TSource, TKey> , Func <TSource, TElement> elementSelector)
- IEnumerable <TSource> , Func <TSource, TKey> keySelector, Func <TSource, TElement> elementSelector, IEqualityComparer <TKey>) <TKey, TElement>
- public static List <TSource> ToList <TSource> (IEnumerable <TSource>)
- public static ILookup <TKey, TSource> ToLookup <TSource, TKey> (IEnumerable <TSource> , Func <TSource, TKey> keySelector)
- public static ILookup <TKey, TSource> ToLookup <TSource, TKey> (IEnumerable <TSource> , Func <TSource, TKey> keySelector, IEqualityComparer <TKey>)

- public static ILookup <TKey, TElement> ToLookup <TSource, TKey, TElement> (IEnumerable <TSource> , Func <TSource, TKey> , Func <TSource, TElement> elementSelector)
- IEnumerable <TSource> , Func <TSource, TKey> keySelector, Func <TSource, TElement> elementSelector, IEqualityComparer <TKey>) ILookup <TKey, TElement> ToLookup <TSource, TKey, TElement>
- public static IEnumerable <TSource> Union <TSource> (IEnumerable <TSource> , IEnumerable <TSource> second)
- public static IEnumerable <TSource> Union <TSource> (IEnumerable <TSource> , IEnumerable <TSource> , IEqualityComparer <TSource>)
- public static IEnumerable <TSource> <TSource> (IEnumerable <TSource> , Func <TSource, Boolean>)
- public static IEnumerable <TSource> <TSource> (IEnumerable <TSource> , Func <TSource, Int32, >)
- public static IEnumerable <TResult> Zip <TFirst, TSecond, TResult> (IEnumerable <TFirst> , IEnumerable <TSecond> , Func <TFirst, TSecond, TResult> resultSelector)

- LINQ .

LINQ `System.Core System.Linq.Enumerable IEnumerable<T>` . .NET Framework 3.5 .

LINQ `IEnumerable` , .

LINQ `List<T> IEnumerable<T>` . LINQ SQL . . LINQ to SQL .

`Contains Except` T `IEquatable<T>.Equals` . , `Equals GetHashCode Object` (override) . `IEqualityComparer<T>` .

`...OrDefault default(T)` .

:

`IEnumerable<T>` . . `.Where()` : `.Where()` `IEnumerable<T>` . . .

( ) `.ToArray()` `.ToList()` . `.ToDictionary()` `.ToLookup()` . .

`ToArray()` `ToList()` ?

`.ToArray()` `.ToList()` `IEnumerable<T>` . .

- API `T[]` `List<T>` .
- `.ToList()` `.ToArray()` . .
- `List<T>` `.ToList()` , , `T[]` `.ToArray()` . , `List<T>` `T[]` .
- `T[]` `.ToArray()` `List<T>` `.ToList()` , `.ToArray()` . `List<T>.TrimExcess()` , `.ToList()` .

Examples

()

```
var persons = new[]
{
    new {Id = 1, Name = "Foo"},
    new {Id = 2, Name = "Bar"},
    new {Id = 3, Name = "Fizz"},
    new {Id = 4, Name = "Buzz"}
};

var names = persons.Select(p => p.Name);
Console.WriteLine(string.Join(",", names.ToArray()));

//Foo,Bar,Fizz,Buzz
```

map .

()

IEnumerable .

```
var personNames = new[]
{
    "Foo", "Bar", "Fizz", "Buzz"
};

var namesStartingWithF = personNames.Where(p => p.StartsWith("F"));
Console.WriteLine(string.Join(",", namesStartingWithF));
```

:

    ,

```
var persons = new[]
{
    new {Id = 1, Name = "Foo"},
    new {Id = 2, Name = "Bar"},
    new {Id = 3, Name = "Fizz"},
    new {Id = 4, Name = "Buzz"}
};

var personsSortedByName = persons.OrderBy(p => p.Name);

Console.WriteLine(string.Join(",", personsSortedByName.Select(p => p.Id).ToArray()));

//2,4,3,1
```

OrderByDescending

```
var persons = new[]
{
    new {Id = 1, Name = "Foo"},
    new {Id = 2, Name = "Bar"},
    new {Id = 3, Name = "Fizz"},
```

```
    new {Id = 4, Name = "Buzz"}
};

var personsSortedByNameDescending = persons.OrderByDescending(p => p.Name);

Console.WriteLine(string.Join(",", personsSortedByNameDescending.Select(p =>
p.Id).ToArray()));

//1,3,4,2
```

```
var numbers = new[] {1,2,3,4,5};
Console.WriteLine(numbers.Contains(3)); //True
Console.WriteLine(numbers.Contains(34)); //False
```

```
var numbers = new[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
var evenNumbersBetweenSixAndFourteen = new[] { 6, 8, 10, 12 };

var result = numbers.Except(evenNumbersBetweenSixAndFourteen);

Console.WriteLine(string.Join(",", result));

//1, 2, 3, 4, 5, 7, 9
```

```
var numbers1to10 = new[] {1,2,3,4,5,6,7,8,9,10};
var numbers5to15 = new[] {5,6,7,8,9,10,11,12,13,14,15};

var numbers5to10 = numbers1to10.Intersect(numbers5to15);

Console.WriteLine(string.Join(",", numbers5to10));

//5,6,7,8,9,10
```

```
var numbers1to5 = new[] {1, 2, 3, 4, 5};
var numbers4to8 = new[] {4, 5, 6, 7, 8};

var numbers1to8 = numbers1to5.Concat(numbers4to8);

Console.WriteLine(string.Join(",", numbers1to8));

//1,2,3,4,5,4,5,6,7,8
```

. Union .

()

```
var numbers = new[] {1,2,3,4,5};

var firstNumber = numbers.First();
Console.WriteLine(firstNumber); //1

var firstEvenNumber = numbers.First(n => (n & 1) == 0);
Console.WriteLine(firstEvenNumber); //2
```

" " InvalidOperationException throw InvalidOperationException .

```
var firstNegativeNumber = numbers.First(n => n < 0);
```

```
var oneNumber = new[] {5};
var theOnlyNumber = oneNumber.Single();
Console.WriteLine(theOnlyNumber);  //5

var numbers = new[] {1,2,3,4,5};

var theOnlyNumberSmallerThanTwo = numbers.Single(n => n < 2);
Console.WriteLine(theOnlyNumberSmallerThanTwo);  //1
```

`InvalidOperationException` throw.

```
var theOnlyNumberInNumbers = numbers.Single();
var theOnlyNegativeNumber = numbers.Single(n => n < 0);
```

```
var numbers = new[] {1,2,3,4,5};

var lastNumber = numbers.Last();
Console.WriteLine(lastNumber); //5

var lastEvenNumber = numbers.Last(n => (n & 1) == 0);
Console.WriteLine(lastEvenNumber); //4
```

`InvalidOperationException` throw `InvalidOperationException` .

```
var lastNegativeNumber = numbers.Last(n => n < 0);
```

LastOrDefault

```
var numbers = new[] {1,2,3,4,5};

var lastNumber = numbers.LastOrDefault();
Console.WriteLine(lastNumber); //5

var lastEvenNumber = numbers.LastOrDefault(n => (n & 1) == 0);
Console.WriteLine(lastEvenNumber); //4

var lastNegativeNumber = numbers.LastOrDefault(n => n < 0);
Console.WriteLine(lastNegativeNumber); //0

var words = new[] { "one", "two", "three", "four", "five" };

var lastWord = words.LastOrDefault();
Console.WriteLine(lastWord); // five

var lastLongWord = words.LastOrDefault(w => w.Length > 4);
Console.WriteLine(lastLongWord); // three

var lastMissingWord = words.LastOrDefault(w => w.Length > 5);
Console.WriteLine(lastMissingWord); // null
```

SingleOrDefault

```
var oneNumber = new[] {5};
var theOnlyNumber = oneNumber.SingleOrDefault();
Console.WriteLine(theOnlyNumber);  //5

var numbers = new[] {1,2,3,4,5};

var theOnlyNumberSmallerThanTwo = numbers.SingleOrDefault(n => n < 2);
Console.WriteLine(theOnlyNumberSmallerThanTwo);  //1

var theOnlyNegativeNumber = numbers.SingleOrDefault(n => n < 0);
Console.WriteLine(theOnlyNegativeNumber);  //0
```

`InvalidOperationException` throw `InvalidOperationException` .

```
var theOnlyNumberInNumbers = numbers.SingleOrDefault();
```

FirstOrDefault

```
var numbers = new[] {1,2,3,4,5};

var firstNumber = numbers.FirstOrDefault();
Console.WriteLine(firstNumber); //1

var firstEvenNumber = numbers.FirstOrDefault(n => (n & 1) == 0);
Console.WriteLine(firstEvenNumber); //2

var firstNegativeNumber = numbers.FirstOrDefault(n => n < 0);
Console.WriteLine(firstNegativeNumber); //0

var words = new[] { "one", "two", "three", "four", "five" };

var firstWord = words.FirstOrDefault();
Console.WriteLine(firstWord); // one

var firstLongWord = words.FirstOrDefault(w => w.Length > 3);
Console.WriteLine(firstLongWord); // three

var firstMissingWord = words.FirstOrDefault(w => w.Length > 5);
Console.WriteLine(firstMissingWord); // null
```

`true` `true` .

```
var numbers = new[] {1,2,3,4,5};

var isNotEmpty = numbers.Any();
Console.WriteLine(isNotEmpty); //True

var anyNumberIsOne = numbers.Any(n => n == 1);
Console.WriteLine(anyNumberIsOne); //True

var anyNumberIsSix = numbers.Any(n => n == 6);
Console.WriteLine(anyNumberIsSix); //False

var anyNumberIsOdd = numbers.Any(n => (n & 1) == 1);
Console.WriteLine(anyNumberIsOdd); //True

var anyNumberIsNegative = numbers.Any(n => n < 0);
```

```
Console.WriteLine(anyNumberIsNegative); //False
```

```
var numbers = new[] {1,2,3,4,5};

var allNumbersAreOdd = numbers.All(n => (n & 1) == 1);
Console.WriteLine(allNumbersAreOdd); //False

var allNumbersArePositive = numbers.All(n => n > 0);
Console.WriteLine(allNumbersArePositive); //True
```

All false . true true .

```
var numbers = new int[0];
var allNumbersArePositive = numbers.All(n => n > 0);
Console.WriteLine(allNumbersArePositive); //True
```

SelectMany ()

Enumerable.Select . Enumerable.SelectMany . .

Enumerable.Select Lambda expressions . Enumerable.SelectMany . .

```
class Invoice
{
    public int Id { get; set; }
}

class Customer
{
    public Invoice[] Invoices {get;set;}
}

var customers = new[] {
    new Customer {
        Invoices = new[] {
            new Invoice {Id=1},
            new Invoice {Id=2},
        }
    },
    new Customer {
        Invoices = new[] {
            new Invoice {Id=3},
            new Invoice {Id=4},
        }
    },
    new Customer {
        Invoices = new[] {
            new Invoice {Id=5},
            new Invoice {Id=6},
        }
    }
};

var allInvoicesFromAllCustomers = customers.SelectMany(c => c.Invoices);

Console.WriteLine(
```

```
    string.Join(",", allInvoicesFromAllCustomers.Select(i => i.Id).ToArray()));
```

:

1,2,3,4,5,6

Enumerable.SelectMany from .

```
var allInvoicesFromAllCustomers
    = from customer in customers
      from invoice in customer.Invoices
      select invoice;
```

```
var numbers = new[] {1,2,3,4};

var sumOfAllNumbers = numbers.Sum();
Console.WriteLine(sumOfAllNumbers); //10

var cities = new[] {
    new {Population = 1000},
    new {Population = 2500},
    new {Population = 4000}
};

var totalPopulation = cities.Sum(c => c.Population);
Console.WriteLine(totalPopulation); //7500
```

Skip N . N + 1 Skip .

```
var numbers = new[] {1,2,3,4,5};

var allNumbersExceptFirstTwo = numbers.Skip(2);
Console.WriteLine(string.Join(",", allNumbersExceptFirstTwo.ToArray()));

//3,4,5
```

n .

```
var numbers = new[] {1,2,3,4,5};

var threeFirstNumbers = numbers.Take(3);
Console.WriteLine(string.Join(",", threeFirstNumbers.ToArray()));

//1,2,3
```

SequenceEqual

```
var numbers = new[] {1,2,3,4,5};
var sameNumbers = new[] {1,2,3,4,5};
var sameNumbersInDifferentOrder = new[] {5,1,4,2,3};

var equalIfSameOrder = numbers.SequenceEqual(sameNumbers);
Console.WriteLine(equalIfSameOrder); //True
```

```
var equalIfDifferentOrder = numbers.SequenceEqual(sameNumbersInDifferentOrder);
Console.WriteLine(equalIfDifferentOrder); //False
```

```
var numbers = new[] {1,2,3,4,5};
var reversed = numbers.Reverse();

Console.WriteLine(string.Join(",", reversed.ToArray()));

//5,4,3,2,1
```

OfType

```
var mixed = new object[] {1,"Foo",2,"Bar",3,"Fizz",4,"Buzz"};
var numbers = mixed.OfType<int>();

Console.WriteLine(string.Join(",", numbers.ToArray()));

//1,2,3,4
```

```
var numbers = new[] {1,2,3,4};

var maxNumber = numbers.Max();
Console.WriteLine(maxNumber); //4

var cities = new[] {
    new {Population = 1000},
    new {Population = 2500},
    new {Population = 4000}
};

var maxPopulation = cities.Max(c => c.Population);
Console.WriteLine(maxPopulation); //4000
```

```
var numbers = new[] {1,2,3,4};

var minNumber = numbers.Min();
Console.WriteLine(minNumber); //1

var cities = new[] {
    new {Population = 1000},
    new {Population = 2500},
    new {Population = 4000}
};

var minPopulation = cities.Min(c => c.Population);
Console.WriteLine(minPopulation); //1000
```

```
var numbers = new[] {1,2,3,4};

var averageNumber = numbers.Average();
Console.WriteLine(averageNumber);
// 2,5
```

.

```
var cities = new[] {
    new {Population = 1000},
    new {Population = 2000},
    new {Population = 4000}
};

var averagePopulation = cities.Average(c => c.Population);
Console.WriteLine(averagePopulation);
// 2333,33
```

.

.NET 4.0

```
var tens = new[] {10,20,30,40,50};
var units = new[] {1,2,3,4,5};

var sums = tens.Zip(units, (first, second) => first + second);

Console.WriteLine(string.Join(",", sums));

//11,22,33,44,55
```

```
var numbers = new[] {1, 1, 2, 2, 3, 3, 4, 4, 5, 5};
var distinctNumbers = numbers.Distinct();

Console.WriteLine(string.Join(",", distinctNumbers));

//1,2,3,4,5
```

GroupBy

```
var persons = new[] {
    new { Name="Fizz", Job="Developer"},
    new { Name="Buzz", Job="Developer"},
    new { Name="Foo", Job="Astronaut"},
    new { Name="Bar", Job="Astronaut"},
};

var groupedByJob = persons.GroupBy(p => p.Job);

foreach(var theGroup in groupedByJob)
{
    Console.WriteLine(
        "{0} are {1}s",
        string.Join(",", theGroup.Select(g => g.Name).ToArray()),
        theGroup.Key);
}

//Fizz,Buzz are Developers
//Foo,Bar are Astronauts
```

,

```
var a = db.Invoices.GroupBy(i => i.Country)
          .Select(g => new { Country = g.Key,
                             Count = g.Count(),
                             Total = g.Sum(i => i.Paid),
                             Average = g.Average(i => i.Paid) });
```

.

```
var a = db.Invoices.GroupBy(i => 1)
          .Select(g => new { Count = g.Count(),
                             Total = g.Sum(i => i.Paid),
                             Average = g.Average(i => i.Paid) });
```

```
var a = db.Invoices.GroupBy(g => 1)
          .Select(g => new { High = g.Count(i => i.Paid >= 1000),
                             Low = g.Count(i => i.Paid < 1000),
                             Sum = g.Sum(i => i.Paid) });
```

ToDictionary

keySelector `IEnumerable` . keySelector injective ( ) `ArgumentException` throw. .

```
var persons = new[] {
    new { Name="Fizz", Id=1},
    new { Name="Buzz", Id=2},
    new { Name="Foo", Id=3},
    new { Name="Bar", Id=4},
};
```

`Dictionary<TKey,TVal>` `TKey` , `TVal` , .

```
var personsById = persons.ToDictionary(p => p.Id);
// personsById is a Dictionary<int,object>

Console.WriteLine(personsById[1].Name); //Fizz
Console.WriteLine(personsById[2].Name); //Buzz
```

`Dictionary<TKey,TVal>` `TKey` `TVal` .

```
var namesById = persons.ToDictionary(p => p.Id, p => p.Name);
//namesById is a Dictionary<int,string>

Console.WriteLine(namesById[3]); //Foo
Console.WriteLine(namesById[4]); //Bar
```

. throw.

```
var persons = new[] {
    new { Name="Fizz", Id=1},
    new { Name="Buzz", Id=2},
    new { Name="Foo", Id=3},
    new { Name="Bar", Id=4},
    new { Name="Oops", Id=4}
```

```
};

var willThrowException = persons.ToDictionary(p => p.Id)
```

ToLookup . , ToLookup ToDictionary Lookup .

```
var numbers1to5 = new[] {1,2,3,4,5};
var numbers4to8 = new[] {4,5,6,7,8};

var numbers1to8 = numbers1to5.Union(numbers4to8);

Console.WriteLine(string.Join(",", numbers1to8));

//1,2,3,4,5,6,7,8
```

. , Concat .

ToArray

```
var numbers = new[] {1,2,3,4,5,6,7,8,9,10};
var someNumbers = numbers.Where(n => n < 6);

Console.WriteLine(someNumbers.GetType().Name);
//WhereArrayIterator`1

var someNumbersArray = someNumbers.ToArray();

Console.WriteLine(someNumbersArray.GetType().Name);
//Int32[]
```

ToList

```
var numbers = new[] {1,2,3,4,5,6,7,8,9,10};
var someNumbers = numbers.Where(n => n < 6);

Console.WriteLine(someNumbers.GetType().Name);
//WhereArrayIterator`1

var someNumbersList = someNumbers.ToList();

Console.WriteLine(
    someNumbersList.GetType().Name + " - " +
    someNumbersList.GetType().GetGenericArguments()[0].Name);
//List`1 - Int32
```

```
IEnumerable<int> numbers = new[] {1,2,3,4,5,6,7,8,9,10};

var numbersCount = numbers.Count();
Console.WriteLine(numbersCount); //10

var evenNumbersCount = numbers.Count(n => (n & 1) == 0);
Console.WriteLine(evenNumbersCount); //5
```

```
var names = new[] {"Foo","Bar","Fizz","Buzz"};

var thirdName = names.ElementAt(2);
Console.WriteLine(thirdName); //Fizz

//The following throws ArgumentOutOfRangeException

var minusOnethName = names.ElementAt(-1);
var fifthName = names.ElementAt(4);
```

ElementAtOrDefault

```
var names = new[] {"Foo","Bar","Fizz","Buzz"};

var thirdName = names.ElementAtOrDefault(2);
Console.WriteLine(thirdName); //Fizz

var minusOnethName = names.ElementAtOrDefault(-1);
Console.WriteLine(minusOnethName); //null

var fifthName = names.ElementAtOrDefault(4);
Console.WriteLine(fifthName); //null
```

SkipWhile

```
var numbers = new[] {2,4,6,8,1,3,5,7};

var oddNumbers = numbers.SkipWhile(n => (n & 1) == 0);

Console.WriteLine(string.Join(",", oddNumbers.ToArray()));

//1,3,5,7
```

TakeWhile

```
var numbers = new[] {2,4,6,1,3,5,7,8};

var evenNumbers = numbers.TakeWhile(n => (n & 1) == 0);

Console.WriteLine(string.Join(",", evenNumbers.ToArray()));

//2,4,6
```

DefaultIfEmpty

```
var numbers = new[] {2,4,6,8,1,3,5,7};

var numbersOrDefault = numbers.DefaultIfEmpty();
Console.WriteLine(numbers.SequenceEqual(numbersOrDefault)); //True

var noNumbers = new int[0];

var noNumbersOrDefault = noNumbers.DefaultIfEmpty();
Console.WriteLine(noNumbersOrDefault.Count()); //1
```

```
Console.WriteLine(noNumbersOrDefault.Single()); //0

var noNumbersOrExplicitDefault = noNumbers.DefaultIfEmpty(34);
Console.WriteLine(noNumbersOrExplicitDefault.Count()); //1
Console.WriteLine(noNumbersOrExplicitDefault.Single()); //34
```

()

:

```
var elements = new[] {1,2,3,4,5};

var commaSeparatedElements = elements.Aggregate(
    seed: "",
    func: (aggregate, element) => $"{aggregate}{element},");

Console.WriteLine(commaSeparatedElements);  //1,2,3,4,5,
```

:

```
var commaSeparatedElements2 = elements.Aggregate(
    seed: new StringBuilder(),
    func: (seed, element) => seed.Append($"{element},"));

Console.WriteLine(commaSeparatedElements2.ToString());  //1,2,3,4,5,
```

:

```
var commaSeparatedElements3 = elements.Aggregate(
    seed: new StringBuilder(),
    func: (seed, element) => seed.Append($"{element},"),
    resultSelector: (seed) => seed.ToString());
Console.WriteLine(commaSeparatedElements3);  //1,2,3,4,5,
```

.

```
var seedAndElements = elements.Select(n=>n.ToString());
var commaSeparatedElements4 = seedAndElements.Aggregate(
    func: (aggregate, element) => $"{aggregate}{element},");

Console.WriteLine(commaSeparatedElements4);  //12,3,4,5,
```

```
var persons = new[] {
    new { Name="Fizz", Job="Developer"},
    new { Name="Buzz", Job="Developer"},
    new { Name="Foo", Job="Astronaut"},
    new { Name="Bar", Job="Astronaut"},
};

var groupedByJob = persons.ToLookup(p => p.Job);

foreach(var theGroup in groupedByJob)
{
    Console.WriteLine(
```

```
        "{0} are {1}s",
        string.Join(",", theGroup.Select(g => g.Name).ToArray()),
        theGroup.Key);
}

//Fizz,Buzz are Developers
//Foo,Bar are Astronauts
```

```
class Developer
{
    public int Id { get; set; }
    public string Name { get; set; }
}

class Project
{
    public int DeveloperId { get; set; }
    public string Name { get; set; }
}

var developers = new[] {
    new Developer {
        Id = 1,
        Name = "Foobuzz"
    },
    new Developer {
        Id = 2,
        Name = "Barfizz"
    }
};

var projects = new[] {
    new Project {
        DeveloperId = 1,
        Name = "Hello World 3D"
    },
    new Project {
        DeveloperId = 1,
        Name = "Super Fizzbuzz Maker"
    },
    new Project {
        DeveloperId = 2,
        Name = "Citizen Kane - The action game"
    },
    new Project {
        DeveloperId = 2,
        Name = "Pro Pong 2016"
    }
};

var denormalized = developers.Join(
    inner: projects,
    outerKeySelector: dev => dev.Id,
    innerKeySelector: proj => proj.DeveloperId,
    resultSelector:
        (dev, proj) => new {
            ProjectName = proj.Name,
            DeveloperName = dev.Name});

foreach(var item in denormalized)
```

```
{
    Console.WriteLine("{0} by {1}", item.ProjectName, item.DeveloperName);
}

//Hello World 3D by Foobuzz
//Super Fizzbuzz Maker by Foobuzz
//Citizen Kane - The action game by Barfizz
//Pro Pong 2016 by Barfizz
```

```
class Developer
{
    public int Id { get; set; }
    public string Name { get; set; }
}

class Project
{
    public int DeveloperId { get; set; }
    public string Name { get; set; }
}

var developers = new[] {
    new Developer {
        Id = 1,
        Name = "Foobuzz"
    },
    new Developer {
        Id = 2,
        Name = "Barfizz"
    }
};

var projects = new[] {
    new Project {
        DeveloperId = 1,
        Name = "Hello World 3D"
    },
    new Project {
        DeveloperId = 1,
        Name = "Super Fizzbuzz Maker"
    },
    new Project {
        DeveloperId = 2,
        Name = "Citizen Kane - The action game"
    },
    new Project {
        DeveloperId = 2,
        Name = "Pro Pong 2016"
    }
};

var grouped = developers.GroupJoin(
    inner: projects,
    outerKeySelector: dev => dev.Id,
    innerKeySelector: proj => proj.DeveloperId,
    resultSelector:
        (dev, projs) => new {
            DeveloperName = dev.Name,
            ProjectNames = projs.Select(p => p.Name).ToArray()});
```

```
foreach(var item in grouped)
{
    Console.WriteLine(
        "{0}'s projects: {1}",
        item.DeveloperName,
        string.Join(", ", item.ProjectNames));
}

//Foobuzz's projects: Hello World 3D, Super Fizzbuzz Maker
//Barfizz's projects: Citizen Kane - The action game, Pro Pong 2016
```

Cast IEnumerable<T> IEnumerable Enumerable . .

ArrayList IEnumerable<T> .

```
var numbers = new ArrayList() {1,2,3,4,5};
Console.WriteLine(numbers.First());
```

.

```
var numbers = new ArrayList() {1,2,3,4,5};
Console.WriteLine(numbers.Cast<int>().First()); //1
```

Cast . InvalidCastException .

```
var numbers = new int[] {1,2,3,4,5};
decimal[] numbersAsDecimal = numbers.Cast<decimal>().ToArray();
```

.

```
var numbers= new int[] {1,2,3,4,5};
decimal[] numbersAsDecimal = numbers.Select(n => (decimal)n).ToArray();
```

int IEnumerable :

```
IEnumerable<int> emptyList = Enumerable.Empty<int>();
```

IEnumerable Type T .

```
Enumerable.Empty<decimal>() == Enumerable.Empty<decimal>(); // This is True
Enumerable.Empty<int>() == Enumerable.Empty<decimal>();     // This is False
```

ThenBy OrderBy .

```
var persons = new[]
{
    new {Id = 1, Name = "Foo", Order = 1},
    new {Id = 1, Name = "FooTwo", Order = 2},
    new {Id = 2, Name = "Bar", Order = 2},
    new {Id = 2, Name = "BarTwo", Order = 1},
    new {Id = 3, Name = "Fizz", Order = 2},
```

```
    new {Id = 3, Name = "FizzTwo", Order = 1},
};

var personsSortedByName = persons.OrderBy(p => p.Id).ThenBy(p => p.Order);

Console.WriteLine(string.Join(",", personsSortedByName.Select(p => p.Name)));
//This will display :
//Foo,FooTwo,BarTwo,Bar,FizzTwo,Fizz
```

`Range` ( ).

```
// prints 1,2,3,4,5,6,7,8,9,10
Console.WriteLine(string.Join(",", Enumerable.Range(1, 10)));

// prints 10,11,12,13,14
Console.WriteLine(string.Join(",", Enumerable.Range(10, 5)));
```

```
class Person
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
}

class Pet
{
    public string Name { get; set; }
    public Person Owner { get; set; }
}

public static void Main(string[] args)
{
    var magnus = new Person { FirstName = "Magnus", LastName = "Hedlund" };
    var terry = new Person { FirstName = "Terry", LastName = "Adams" };

    var barley = new Pet { Name = "Barley", Owner = terry };

    var people = new[] { magnus, terry };
    var pets = new[] { barley };

    var query =
        from person in people
        join pet in pets on person equals pet.Owner into gj
        from subpet in gj.DefaultIfEmpty()
        select new
        {
            person.FirstName,
            PetName = subpet?.Name ?? "-" // Use - if he has no pet
        };

    foreach (var p in query)
        Console.WriteLine($"{p.FirstName}: {p.PetName}");
}
```

`Enumerable.Repeat` . "Hello" 4 .

```
var repeats = Enumerable.Repeat("Hello", 4);
```

```
foreach (var item in repeats)
{
    Console.WriteLine(item);
}

/* output:
    Hello
    Hello
    Hello
    Hello
*/
```

LINQ : https://riptutorial.com/ko/dot-net/topic/34/linq

15: NuGet

NuGet.org :

NuGet .NET Microsoft . NuGet . NuGet .

NuGet.org .

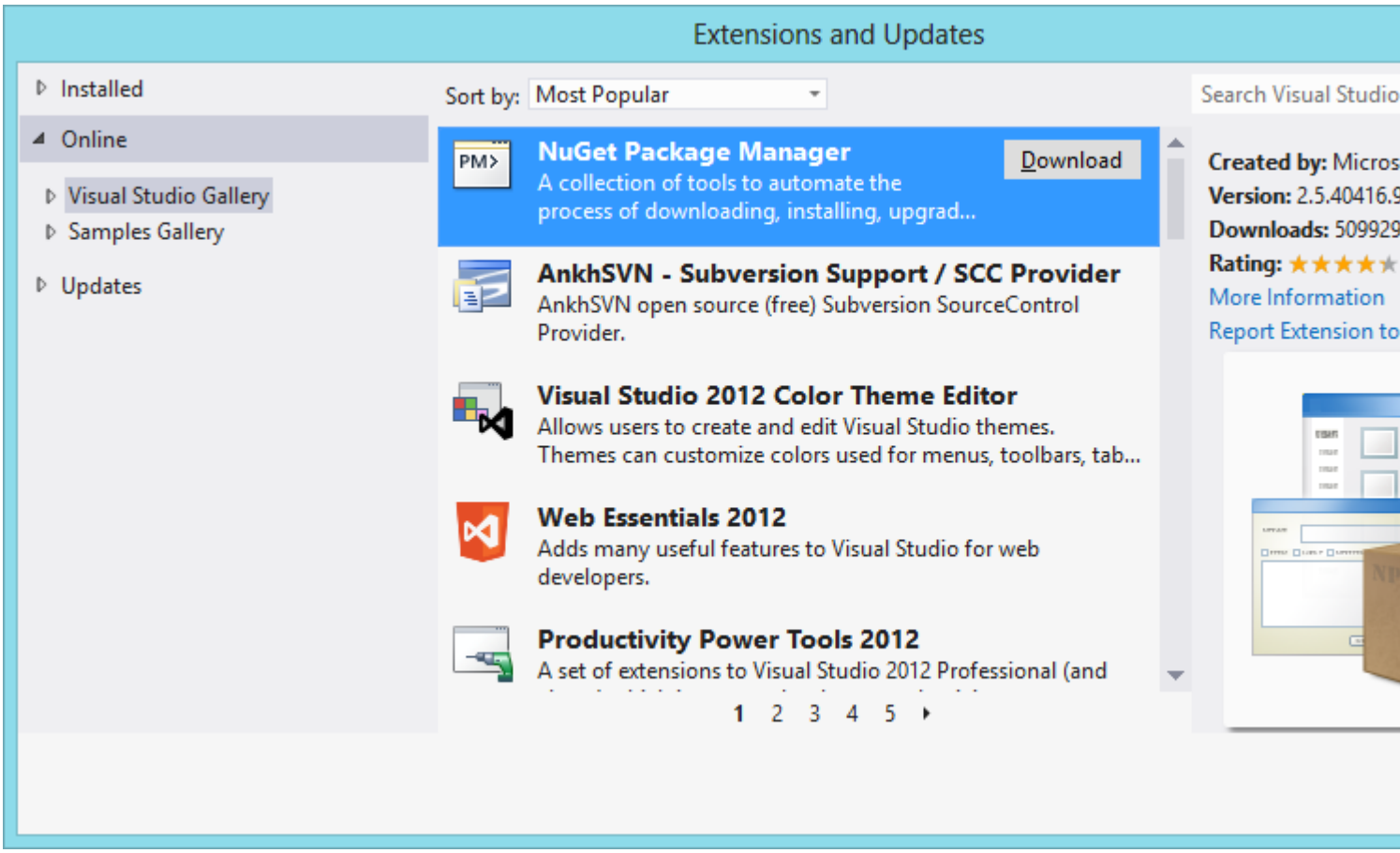
Examples

NuGet

NuGet Package Manager . Visual Studio Extension NuGet Client .

Visual Studio 2012 NuGet -> NuGet -> .

Visual Studio Tools Extensions and Updates .



GUI .

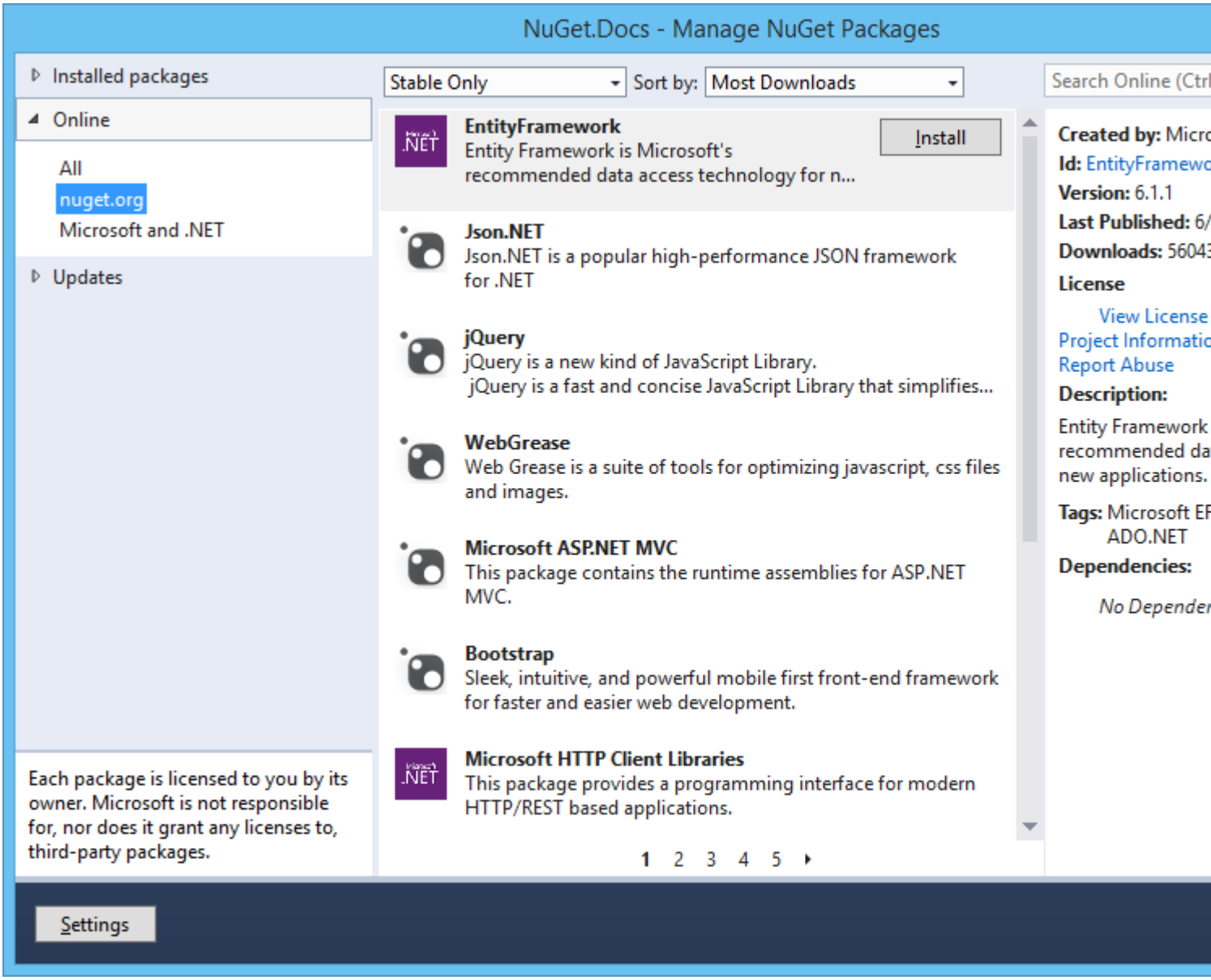
- References "Manage NuGet Packages ..." .

:

- -> NuGet -> .

UI

(References) "Manage NuGet Packages ..." . Package Manager .



Tools -> NuGet Package Manager -> Package Manager Console IDE . .

"Default project" `install-package` .

```
Install-Package Elmah
```

" " .

```
Install-Package Elmah -ProjectName MyFirstWebsite
```

.

```
PM> Update-Package EntityFramework
```

EntityFramework . " " `Install-Package EntityFramework` .

.

```
PM> Update-Package EntityFramework -ProjectName MyFirstWebsite
```

```
PM> Uninstall-Package EntityFramework
```

```
PM> Uninstall-Package -ProjectName MyProjectB EntityFramework
```

```
PM> Install-Package EntityFramework -Version 6.1.2
```

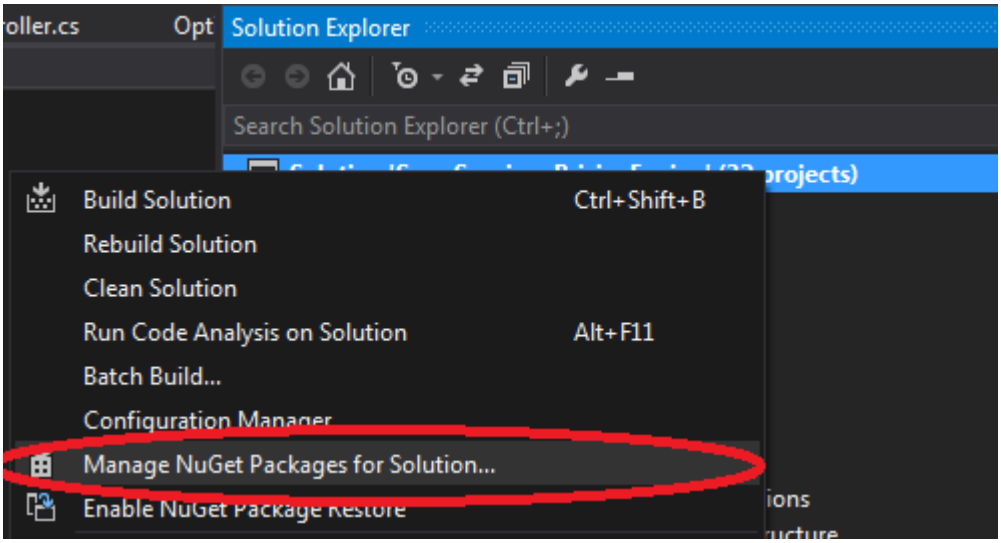
(MyGet, Klondike)

```
nuget sources add -name feedname -source http://sourcefeedurl
```

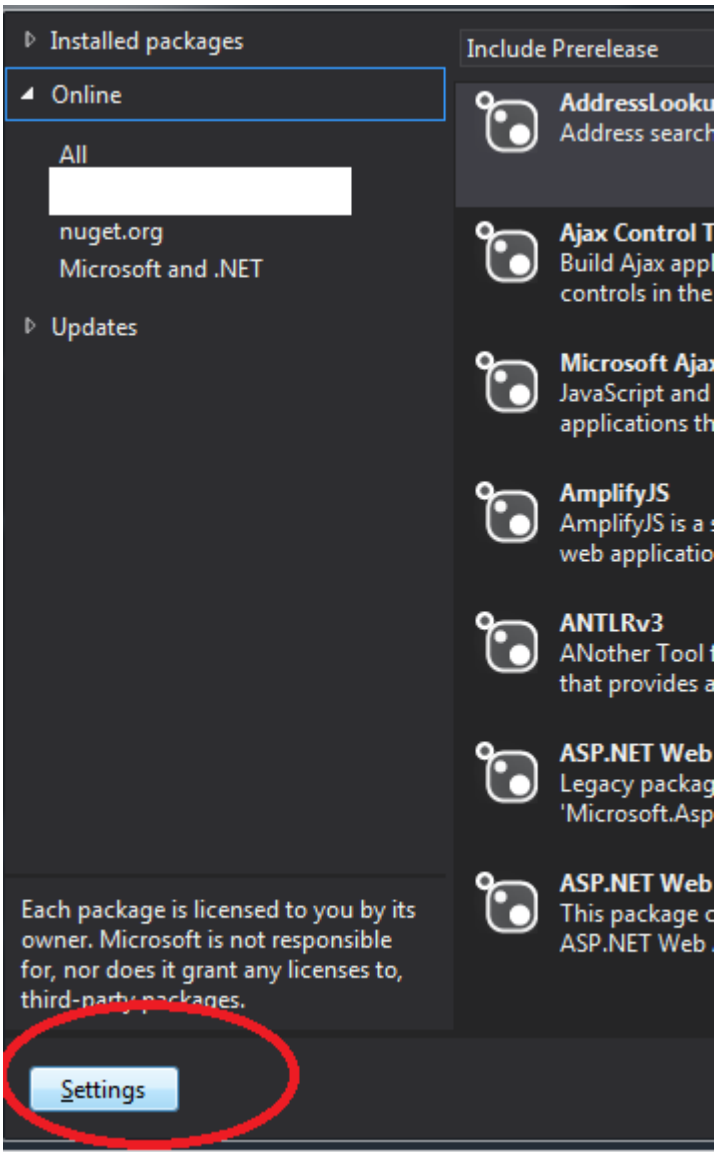
UI () Nuget

.

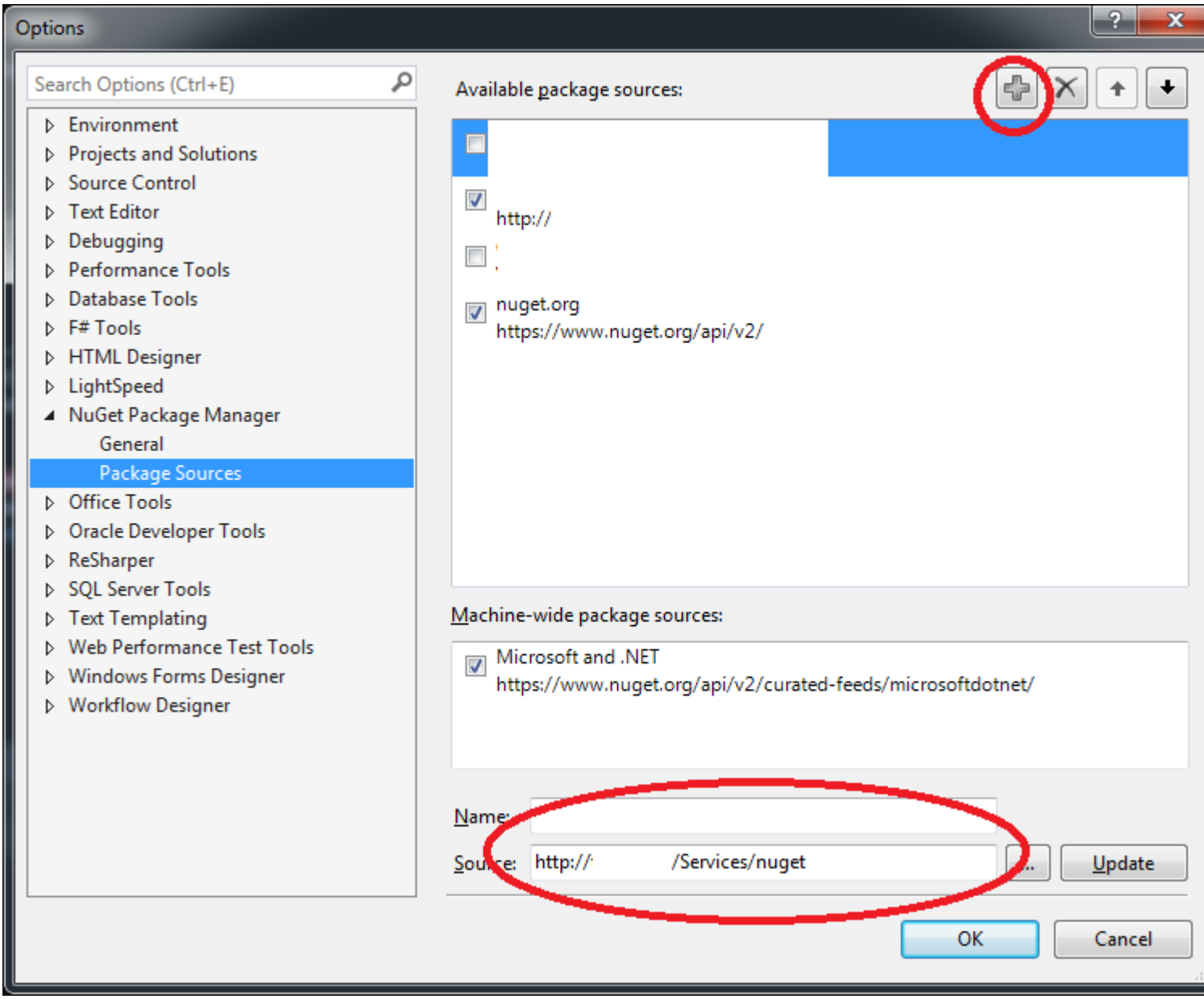
1. □□□ □□□ `Manage NuGet Packages for Solution` Manage NuGet Packages for Solution



2. `Settings` .


Installed packages
Online
All
nuget.org
Microsoft and .NET
Updates
Include Prerelease
AddressLooku
Address search
Ajax Control T
Build Ajax appl
controls in the
Microsoft Ajax
JavaScript and
applications th
AmplifyJS
AmplifyJS is a
web applicatio
ANTLRv3
ANother Tool
that provides a
ASP.NET Web
Legacy packag
'Microsoft.Asp
ASP.NET Web
This package c
ASP.NET Web
Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.
Settings

3. + URL .




```
PM> uninstall-Package EntityFramework -Version 6.1.2
```

NuGet : https://riptutorial.com/ko/dot-net/topic/43/nuget--

16: ReadOnlyCollections

ReadOnlyCollection (') .

ReadOnlyCollection ReadOnlyCollection . ReadOnlyCollection .

ReadOnlyCollection .

ReadOnlyCollection .

- ObservableCollection<T>
- ReadOnlyObservableCollection<T>

ReadOnlyCollections ImmutableCollection

ReadOnlyCollection ImmutableCollection ImmutableCollection . n . ReadOnlyCollection / / .

Examples

ReadOnlyCollection

ReadOnlyCollection IList .

```
var groceryList = new List<string> { "Apple", "Banana" };
var readOnlyGroceryList = new ReadOnlyCollection<string>(groceryList);
```

LINQ

LINQ IList AsReadOnly() .

```
var readOnlyVersion = groceryList.AsReadOnly();
```

ReadOnlyCollection . ReadOnlyCollection .

```
var readOnlyGroceryList = new List<string> {"Apple", "Banana"}.AsReadOnly();
// Great, but you will not be able to update the grocery list because
//  you do not have a reference to the source list anymore!
```

ImmutableCollection .

ReadOnlyCollection

ReadOnlyCollection . ReadOnlyCollection . ReadOnlyCollection .

```
var groceryList = new List<string> { "Apple", "Banana" };

var readOnlyGroceryList = new ReadOnlyCollection<string>(groceryList);

var itemCount = readOnlyGroceryList.Count;  // There are currently 2 items

//readOnlyGroceryList.Add("Candy");         // Compiler Error - Items cannot be added to a
ReadOnlyCollection object
groceryList.Add("Vitamins");                // ..but they can be added to the original
collection

itemCount = readOnlyGroceryList.Count;      // Now there are 3 items
var lastItem = readOnlyGroceryList.Last();  // The last item on the read only list is now
"Vitamins"
```

: ReadOnlyCollection .

ReadOnlyCollection .

```
public class Item
{
    public string Name { get; set; }
    public decimal Price { get; set; }
}

public static void FillOrder()
{
    // An order is generated
    var order = new List<Item>
    {
        new Item { Name = "Apple", Price = 0.50m },
        new Item { Name = "Banana", Price = 0.75m },
        new Item { Name = "Vitamins", Price = 5.50m }
    };

    // The current sub total is $6.75
    var subTotal = order.Sum(item => item.Price);

    // Let the customer preview their order
    var customerPreview = new ReadOnlyCollection<Item>(order);

    // The customer can't add or remove items, but they can change
    //   the price of an item, even though it is a ReadOnlyCollection
    customerPreview.Last().Price = 0.25m;

    // The sub total is now only $1.50!
    subTotal = order.Sum(item => item.Price);
}
```

ReadOnlyCollections : https://riptutorial.com/ko/dot-net/topic/6906/readonlycollections

17: StdErr

Examples

.

```
var sourceFileName = "NonExistingFile";
try
{
    System.IO.File.Copy(sourceFileName, "DestinationFile");
}
catch (Exception e)
{
    var stdErr = Console.Error;
    stdErr.WriteLine($"Failed to copy '{sourceFileName}': {e.Message}");
}
```

```
var errors = new System.Text.StringBuilder();
var process = new Process
{
    StartInfo = new ProcessStartInfo
    {
        RedirectStandardError = true,
        FileName = "xcopy.exe",
        Arguments = "\"NonExistingFile\" \"DestinationFile\"",
        UseShellExecute = false
    },

};
process.ErrorDataReceived += (s, e) => errors.AppendLine(e.Data);
process.Start();
process.BeginErrorReadLine();
process.WaitForExit();

if (errors.Length > 0) // something went wrong
    System.Console.Error.WriteLine($"Child process error: \r\n {errors}");
```

StdErr : https://riptutorial.com/ko/dot-net/topic/10779/stderr----

18: System.IO

Examples

StreamReader

```
string fullOrRelativePath = "testfile.txt";

string fileData;

using (var reader = new StreamReader(fullOrRelativePath))
{
    fileData = reader.ReadToEnd();
}
```

StreamReader .

System.IO.File , File.ReadAllText(path) File.ReadAllLines(path) .

System.IO.File /

.

```
string fileText = File.ReadAllText(file);
string[] fileLines = File.ReadAllLines(file);
byte[] fileBytes = File.ReadAllBytes(file);
```

- .
- . .
- .

. .

```
 File.AppendAllText(file, "Here is some data that is\nappended to the file.");
 File.AppendAllLines(file, new string[2] { "Here is some data that is", "appended to the
file." });
 using (StreamWriter stream = File.AppendText(file))
 {
     stream.WriteLine("Here is some data that is");
     stream.Write("appended to the file.");
 }
```

- .
- .
- File.AppendText File.AppendText . File.AppendText .

. - . .

```
File.WriteAllText(file, "here is some data\nin this file.");
File.WriteAllLines(file, new string[2] { "here is some data", "in this file" });
File.WriteAllBytes(file, new byte[2] { 0, 255 });
```

- .
- .
- .

System.IO.SerialPorts

```
using System.IO.Ports;
string[] ports = SerialPort.GetPortNames();
for (int i = 0; i < ports.Length; i++)
{
    Console.WriteLine(ports[i]);
}
```

System.IO.SerialPort

```
using System.IO.Ports;
SerialPort port = new SerialPort();
SerialPort port = new SerialPort("COM 1"); ;
SerialPort port = new SerialPort("COM 1", 9600);
```

: SerialPort 7 .

SerialPort /

`SerialPort.Read` `SerialPort.Write` . SerialPort `System.IO.Stream` . `SerialPort.BaseStream` .

```
int length = port.BytesToRead;
//Note that you can swap out a byte-array for a char-array if you prefer.
byte[] buffer = new byte[length];
port.Read(buffer, 0, length);
```

.

```
string curData = port.ReadExisting();
```

.

```
string line = port.ReadLine();
```

SerialPort .

```
port.Write("here is some text to be sent over the serial port.");
```

.

```
//Note that you can swap out the byte-array with a char-array if you so choose.
byte[] data = new byte[1] { 255 };
port.Write(data, 0, data.Length);
```

System.IO : https://riptutorial.com/ko/dot-net/topic/5259/system-io

19: System.IO.File

- ;
- ;

source	.
destination	source (( )).

Examples

( ) .

```
File.Delete(path);
```

.

- (UnauthorizedAccessException Throw).
- (IOException).
- (IOException) .
- (IOException).

( ) .

```
if (File.Exists(path))
    File.Delete(path);
```

File.Exists() File.Delete() . I / O ( ).

```
if (File.Exists(path))
{
    try
    {
        File.Delete(path);
    }
    catch (IOException exception)
    {
        if (!File.Exists(path))
            return; // Someone else deleted this file

        // Something went wrong...
    }
    catch (UnauthorizedAccessException exception)
    {
        // I do not have required permissions
    }
}
```

I / O (:) . I / O .

```
public static void Delete(string path)
{
    if (!File.Exists(path))
        return;

    for (int i=1; ; ++i)
    {
        try
        {
            File.Delete(path);
            return;
        }
        catch (IOException e)
        {
            if (!File.Exists(path))
                return;

            if (i == NumberOfAttempts)
                throw;

            Thread.Sleep(DelayBetweenEachAttempt);
        }

        // You may handle UnauthorizedAccessException but this issue
        // will probably won't be fixed in few seconds...
    }
}

private const int NumberOfAttempts = 3;
private const int DelayBetweenEachAttempt = 1000; // ms
```

: Windows . `FileShare.Delete` .

. .

.

```
File.WriteAllLines(path,
    File.ReadAllLines(path).Where(x => !String.IsNullOrWhiteSpace(x)));
```

```
File.WriteAllLines(outputPath,
    File.ReadLines(inputPath).Where(x => !String.IsNullOrWhiteSpace(x)));
```

( ) . ( ).

```
public static void ConvertEncoding(string path, Encoding from, Encoding to)
{
    File.WriteAllText(path, File.ReadAllText(path, from), to);
}
```

BOM (Byte Order Mark) . Encoding.UTF8.GetString Preamble / BOM .

""( )

(System.IO.Directory.GetFiles() System.IO.Directory.EnumerateFiles). ("*.*").

```
public static void Touch(string path,
                         string searchPattern = "*.*",
                         SearchOptions options = SearchOptions.None)
{
    var now = DateTime.Now;

    foreach (var filePath in Directory.EnumerateFiles(path, searchPattern, options))
    {
        File.SetLastWriteTime(filePath, now);
    }
}
```

.

. .

```
static IEnumerable<string> EnumerateAllFilesOlderThan(
                               TimeSpan maximumAge,
                               string path,
                               string searchPattern = "*.*",
                               SearchOption options = SearchOption.TopDirectoryOnly)
{
    DateTime oldestWriteTime = DateTime.Now - maximumAge;

    return Directory.EnumerateFiles(path, searchPattern, options)
        .Where(x => Directory.GetLastWriteTime(x) < oldestWriteTime);
}
```

:

```
var oldFiles = EnumerateAllFilesOlderThan(TimeSpan.FromDays(7), @"c:\log", "*.log");
```

:

- Directory.EnumerateFiles() Directory.GetFiles() . . .
- (:).
- (, ,) MSDN .

File.Move

.

```
File.Move(@"C:\TemporaryFile.txt", @"C:\TemporaryFiles\TemporaryFile.txt");
```

. , 'C' ? - 'B' 'M' ?

( ) ?

.

```
string source = @"C:\TemporaryFile.txt", destination = @"C:\TemporaryFiles\TemporaryFile.txt";
if(File.Exists("C:\TemporaryFile.txt"))
{
    File.Move(source, destination);
}
```

. File.Exists . .

FileNotFoundException .

.

IOException	.
ArgumentNullException	Source / Destination null.
ArgumentException	/ .
UnauthorizedAccessException	.
PathTooLongException	, . Windows 248 260 .
DirectoryNotFoundException	.
NotSupportedException	.

System.IO.File : https://riptutorial.com/ko/dot-net/topic/5395/system-io-file-

20: System.Net.Mail

System.Net.MailMessage . using Disposable Exceptions .

Examples

MailMessage

. `SmtpClient` . 25 .

```
public class clsMail
{
    private static bool SendMail(string mailfrom, List<string>replytos, List<string> mailtos,
List<string> mailccs, List<string> mailbccs, string body, string subject, List<string>
Attachment)
    {
        try
        {
            using(MailMessage MyMail = new MailMessage())
            {
                MyMail.From = new MailAddress(mailfrom);
                foreach (string mailto in mailtos)
                    MyMail.To.Add(mailto);

                if (replytos != null && replytos.Any())
                {
                    foreach (string replyto in replytos)
                        MyMail.ReplyToList.Add(replyto);
                }

                if (mailccs != null && mailccs.Any())
                {
                    foreach (string mailcc in mailccs)
                        MyMail.CC.Add(mailcc);
                }

                if (mailbccs != null && mailbccs.Any())
                {
                    foreach (string mailbcc in mailbccs)
                        MyMail.Bcc.Add(mailbcc);
                }

                MyMail.Subject = subject;
                MyMail.IsBodyHtml = true;
                MyMail.Body = body;
                MyMail.Priority = MailPriority.Normal;

                if (Attachment != null && Attachment.Any())
                {
                    System.Net.Mail.Attachment attachment;
                    foreach (var item in Attachment)
                    {
                        attachment = new System.Net.Mail.Attachment(item);
                        MyMail.Attachments.Add(attachment);
                    }
                }
```

```
                SmtpClient smtpMailObj = new SmtpClient();
                smtpMailObj.Host = "your host";
                smtpMailObj.Port = 25;
                smtpMailObj.Credentials = new System.Net.NetworkCredential("uid", "pwd");

                smtpMailObj.Send(MyMail);
                return true;
            }
        }
        catch
        {
            return false;
        }
    }
}
```

MailMessage SmtpClient . () .

```
using System.Net.Mail;

using(MailMessage myMail = new MailMessage())
{
     Attachment attachment = new Attachment(path);
     myMail.Attachments.Add(attachment);

     // further processing to send the mail message

}
```

System.Net.Mail : https://riptutorial.com/ko/dot-net/topic/7440/system-net-mail

21: System.Reflection.Emit

Examples

```
using System;
using System.Reflection;
using System.Reflection.Emit;

class DemoAssemblyBuilder
{
    public static void Main()
    {
        // An assembly consists of one or more modules, each of which
        // contains zero or more types. This code creates a single-module
        // assembly, the most common case. The module contains one type,
        // named "MyDynamicType", that has a private field, a property
        // that gets and sets the private field, constructors that
        // initialize the private field, and a method that multiplies
        // a user-supplied number by the private field value and returns
        // the result. In C# the type might look like this:
        /*
        public class MyDynamicType
        {
            private int m_number;

            public MyDynamicType() : this(42) {}
            public MyDynamicType(int initNumber)
            {
                m_number = initNumber;
            }

            public int Number
            {
                get { return m_number; }
                set { m_number = value; }
            }

            public int MyMethod(int multiplier)
            {
                return m_number * multiplier;
            }
        }
        */

        AssemblyName aName = new AssemblyName("DynamicAssemblyExample");
        AssemblyBuilder ab =
            AppDomain.CurrentDomain.DefineDynamicAssembly(
                aName,
                AssemblyBuilderAccess.RunAndSave);

        // For a single-module assembly, the module name is usually
        // the assembly name plus an extension.
        ModuleBuilder mb =
            ab.DefineDynamicModule(aName.Name, aName.Name + ".dll");

        TypeBuilder tb = mb.DefineType(
            "MyDynamicType",
```

```
             TypeAttributes.Public);

        // Add a private field of type int (Int32).
        FieldBuilder fbNumber = tb.DefineField(
            "m_number",
            typeof(int),
            FieldAttributes.Private);

        // Next, we make a simple sealed method.
        MethodBuilder mbMyMethod = tb.DefineMethod(
            "MyMethod",
            MethodAttributes.Public,
            typeof(int),
            new[] { typeof(int) });

        ILGenerator il = mbMyMethod.GetILGenerator();
        il.Emit(OpCodes.Ldarg_0); // Load this - always the first argument of any instance
method
        il.Emit(OpCodes.Ldfld, fbNumber);
        il.Emit(OpCodes.Ldarg_1); // Load the integer argument
        il.Emit(OpCodes.Mul); // Multiply the two numbers with no overflow checking
        il.Emit(OpCodes.Ret); // Return

        // Next, we build the property. This involves building the property itself, as well as
the
        // getter and setter methods.
        PropertyBuilder pbNumber = tb.DefineProperty(
            "Number", // Name
            PropertyAttributes.None,
            typeof(int), // Type of the property
            new Type[0]); // Types of indices, if any

        MethodBuilder mbSetNumber = tb.DefineMethod(
            "set_Number", // Name - setters are set_Property by convention
            // Setter is a special method and we don't want it to appear to callers from C#
            MethodAttributes.PrivateScope | MethodAttributes.HideBySig |
MethodAttributes.Public | MethodAttributes.SpecialName,
            typeof(void), // Setters don't return a value
            new[] { typeof(int) }); // We have a single argument of type System.Int32

        // To generate the body of the method, we'll need an IL generator
        il = mbSetNumber.GetILGenerator();
        il.Emit(OpCodes.Ldarg_0); // Load this
        il.Emit(OpCodes.Ldarg_1); // Load the new value
        il.Emit(OpCodes.Stfld, fbNumber); // Save the new value to this.m_number
        il.Emit(OpCodes.Ret); // Return

        // Finally, link the method to the setter of our property
        pbNumber.SetSetMethod(mbSetNumber);

        MethodBuilder mbGetNumber = tb.DefineMethod(
            "get_Number",
            MethodAttributes.PrivateScope | MethodAttributes.HideBySig |
MethodAttributes.Public | MethodAttributes.SpecialName,
            typeof(int),
            new Type[0]);

        il = mbGetNumber.GetILGenerator();
        il.Emit(OpCodes.Ldarg_0); // Load this
        il.Emit(OpCodes.Ldfld, fbNumber); // Load the value of this.m_number
        il.Emit(OpCodes.Ret); // Return the value
```

```
        pbNumber.SetGetMethod(mbGetNumber);

        // Finally, we add the two constructors.
        // Constructor needs to call the constructor of the parent class, or another
constructor in the same class
        ConstructorBuilder intConstructor = tb.DefineConstructor(
            MethodAttributes.Public, CallingConventions.Standard | CallingConventions.HasThis,
new[] { typeof(int) });
        il = intConstructor.GetILGenerator();
        il.Emit(OpCodes.Ldarg_0); // this
        il.Emit(OpCodes.Call, typeof(object).GetConstructor(new Type[0])); // call parent's
constructor
        il.Emit(OpCodes.Ldarg_0); // this
        il.Emit(OpCodes.Ldarg_1); // our int argument
        il.Emit(OpCodes.Stfld, fbNumber); // store argument in this.m_number
        il.Emit(OpCodes.Ret);

        var parameterlessConstructor = tb.DefineConstructor(
            MethodAttributes.Public, CallingConventions.Standard | CallingConventions.HasThis,
new Type[0]);
        il = parameterlessConstructor.GetILGenerator();
        il.Emit(OpCodes.Ldarg_0); // this
        il.Emit(OpCodes.Ldc_I4_S, (byte)42); // load 42 as an integer constant
        il.Emit(OpCodes.Call, intConstructor); // call this(42)
        il.Emit(OpCodes.Ret);

        // And make sure the type is created
        Type ourType = tb.CreateType();

        // The types from the assembly can be used directly using reflection, or we can save
the assembly to use as a reference
        object ourInstance = Activator.CreateInstance(ourType);
        Console.WriteLine(ourType.GetProperty("Number").GetValue(ourInstance)); // 42

        // Save the assembly for use elsewhere. This is very useful for debugging - you can
use e.g. ILSpy to look at the equivalent IL/C# code.
        ab.Save(@"DynamicAssemblyExample.dll");
        // Using newly created type
        var myDynamicType = tb.CreateType();
        var myDynamicTypeInstance = Activator.CreateInstance(myDynamicType);

        Console.WriteLine(myDynamicTypeInstance.GetType()); // MyDynamicType

        var numberField = myDynamicType.GetField("m_number", BindingFlags.NonPublic |
BindingFlags.Instance);
        numberField.SetValue (myDynamicTypeInstance, 10);

        Console.WriteLine(numberField.GetValue(myDynamicTypeInstance)); // 10
    }
}
```

System.Reflection.Emit : https://riptutorial.com/ko/dot-net/topic/74/system-reflection-emit--

22: System.Runtime.Caching.MemoryCache (ObjectCache)

Examples

()

Set CacheItem .

```
ObjectCache.Set(CacheItem, CacheItemPolicy) ObjectCache.Set(CacheItem, CacheItemPolicy)
```

```
private static bool SetToCache()
{
    string key = "Cache_Key";
    string value = "Cache_Value";

    //Get a reference to the default MemoryCache instance.
    var cacheContainer = MemoryCache.Default;

    var policy = new CacheItemPolicy()
    {
        AbsoluteExpiration = DateTimeOffset.Now.AddMinutes(DEFAULT_CACHE_EXPIRATION_MINUTES)
     };
     var itemToCache = new CacheItem(key, value); //Value is of type object.
     cacheContainer.Set(itemToCache, policy);
}
```

System.Runtime.Caching.MemoryCache (ObjectCache)

valueFetchFactory .

```
    public static TValue GetExistingOrAdd<TValue>(string key, double minutesForExpiration,
Func<TValue> valueFetchFactory)
    {
        try
        {
            //The Lazy class provides Lazy initialization which will evaluate
            //the valueFetchFactory only if item is not in the cache.
            var newValue = new Lazy<TValue>(valueFetchFactory);

            //Setup the cache policy if item will be saved back to cache.
            CacheItemPolicy policy = new CacheItemPolicy()
            {
                AbsoluteExpiration = DateTimeOffset.Now.AddMinutes(minutesForExpiration)
            };

            //returns existing item form cache or add the new value if it does not exist.
            var cachedItem = _cacheContainer.AddOrGetExisting(key, newValue, policy) as
Lazy<TValue>;

            return (cachedItem ?? newValue).Value;
        }
```

```
        catch (Exception excep)
        {
            return default(TValue);
        }
    }
```

System.Runtime.Caching.MemoryCache (ObjectCache) : https://riptutorial.com/ko/dot-net/topic/76/system-runtime-caching-memorycache--objectcache-

23: TPL

System.Threading.Tasks.Dataflow

System.Threading.Tasks

System.Net.Http

System.Net

Post SendAsync

Post SendAsync .

Post bool . BoundedCapcity . SendAsync await Task<bool> . (:) false true .

Examples

ActionBlock .

```
// Create a block with an asynchronous action
var block = new ActionBlock<string>(async hostName =>
{
    IPAddress[] ipAddresses = await Dns.GetHostAddressesAsync(hostName);
    Console.WriteLine(ipAddresses[0]);
});

block.Post("google.com"); // Post items to the block's InputQueue for processing
block.Post("reddit.com");
block.Post("stackoverflow.com");

block.Complete(); // Tell the block to complete and stop accepting new items
await block.Completion; // Asynchronously wait until all items completed processingu
```

```
var httpClient = new HttpClient();

// Create a block the accepts a uri and returns its contents as a string
var downloaderBlock = new TransformBlock<string, string>(
    async uri => await httpClient.GetStringAsync(uri));

// Create a block that accepts the content and prints it to the console
var printerBlock = new ActionBlock<string>(
    contents => Console.WriteLine(contents));

// Make the downloaderBlock complete the printerBlock when its completed.
var dataflowLinkOptions = new DataflowLinkOptions {PropagateCompletion = true};

// Link the block to create a pipeline
downloaderBlock.LinkTo(printerBlock, dataflowLinkOptions);
```

```
// Post urls to the first block which will pass their contents to the second one.
downloaderBlock.Post("http://youtube.com");
downloaderBlock.Post("http://github.com");
downloaderBlock.Post("http://twitter.com");

downloaderBlock.Complete(); // Completion will propagate to printerBlock
await printerBlock.Completion; // Only need to wait for the last block in the pipeline
```

BufferBlock /

```
public class Producer
{
    private static Random random = new Random((int)DateTime.UtcNow.Ticks);
    //produce the value that will be posted to buffer block
    public double Produce ( )
    {
        var value = random.NextDouble();
        Console.WriteLine($"Producing value: {value}");
        return value;
    }
}

public class Consumer
{
    //consume the value that will be received from buffer block
    public void Consume (double value) => Console.WriteLine($"Consuming value: {value}");
}

class Program
{
    private static BufferBlock<double> buffer = new BufferBlock<double>();
    static void Main (string[] args)
    {
        //start a task that will every 1 second post a value from the producer to buffer block
        var producerTask = Task.Run(async () =>
        {
            var producer = new Producer();
            while(true)
            {
                buffer.Post(producer.Produce());
                await Task.Delay(1000);
            }
        });
        //start a task that will recieve values from bufferblock and consume it
        var consumerTask = Task.Run(() =>
        {
            var consumer = new Consumer();
            while(true)
            {
                consumer.Consume(buffer.Receive());
            }
        });

        Task.WaitAll(new[] { producerTask, consumerTask });
    }
}
```

Bounded BufferBlock

```
var bufferBlock = new BufferBlock<int>(new DataflowBlockOptions
{
    BoundedCapacity = 1000
});

var cancellationToken = new CancellationTokenSource(TimeSpan.FromSeconds(10)).Token;

var producerTask = Task.Run(async () =>
{
    var random = new Random();

    while (!cancellationToken.IsCancellationRequested)
    {
        var value = random.Next();
        await bufferBlock.SendAsync(value, cancellationToken);
    }
});

var consumerTask = Task.Run(async () =>
{
    while (await bufferBlock.OutputAvailableAsync())
    {
        var value = bufferBlock.Receive();
        Console.WriteLine(value);
    }
});

await Task.WhenAll(producerTask, consumerTask);
```

TPL : https://riptutorial.com/ko/dot-net/topic/784/tpl--

24: VB

Examples

VB.NET Forms Hello World

.

```
Public Class Form1
    Private Sub Form1_Shown(sender As Object, e As EventArgs) Handles MyBase.Shown
        MessageBox.Show("Hello, World!")
    End Sub
End Class
To show a message box before the form has been shown:

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        MessageBox.Show("Hello, World!")
    End Sub
End Class
```

Load () . Show () . Activate () .

Show () Load () . show msgBox () Load () msgBox () . **Load (), Show ()** **.**

VB . :

.

```
'can be permanently set
' Tools / Options / Projects and Soluntions / VB Defaults
Option Strict On
Option Explicit On
Option Infer Off

Public Class Form1

End Class
```

&, + . .

.

Application.DoEvents . " . , .

.

Windows.Forms.Timer . , Timer .

.

```
'can be permanently set
' Tools / Options / Projects and Soluntions / VB Defaults
Option Strict On
Option Explicit On
Option Infer Off

Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Button1.Enabled = False
        Timer1.Interval = 60 * 1000 'one minute intervals
        'start timer
        Timer1.Start()
        Label1.Text = DateTime.Now.ToLongTimeString
    End Sub

    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
        Label1.Text = DateTime.Now.ToLongTimeString
    End Sub
End Class
```

. . 3 . .

```
'can be permanently set
' Tools / Options / Projects and Soluntions / VB Defaults
Option Strict On
Option Explicit On
Option Infer Off

Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Button1.Enabled = False
        ctSecs = 0 'clear count
        Timer1.Interval = 1000 'one second in ms.
        'start timers
        stpw.Reset()
        stpw.Start()
        Timer1.Start()
    End Sub

    Dim stpw As New Stopwatch
    Dim ctSecs As Integer

    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
        ctSecs += 1
        If ctSecs = 180 Then 'about 2.5 seconds off on my PC!
            'stop timing
            stpw.Stop()
            Timer1.Stop()
            'show actual elapsed time
            'Is it near 180?
            Label1.Text = stpw.Elapsed.TotalSeconds.ToString("n1")
        End If
    End Sub
End Class
```

button1 3 label1 . label1 180 ? . 182.5 !

"Windows Forms Timer 55 ." . .

.

```
'can be permanently set
' Tools / Options / Projects and Soluntions / VB Defaults
Option Strict On
Option Explicit On
Option Infer Off

Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Button1.Enabled = False
        Timer1.Interval = 100 'one tenth of a second in ms.
        'start timers
        stpw.Reset()
        stpw.Start()
        Timer1.Start()
    End Sub

    Dim stpw As New Stopwatch
    Dim threeMinutes As TimeSpan = TimeSpan.FromMinutes(3)

    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
        If stpw.Elapsed >= threeMinutes Then '0.1 off on my PC!
            'stop timing
            stpw.Stop()
            Timer1.Stop()
            'show actual elapsed time
            'how close?
            Label1.Text = stpw.Elapsed.TotalSeconds.ToString("n1")
        End If
    End Sub
End Class
```

. .

VB : https://riptutorial.com/ko/dot-net/topic/2197/vb-

25: XmlSerializer

`XmlSerializer` `HTML` . HTML Agility Pack

Examples

```
public void SerializeFoo(string fileName, Foo foo)
{
    var serializer = new XmlSerializer(typeof(Foo));
    using (var stream = File.Open(fileName, FileMode.Create))
    {
        serializer.Serialize(stream, foo);
    }
}
```

```
public Foo DeserializeFoo(string fileName)
{
    var serializer = new XmlSerializer(typeof(Foo));
    using (var stream = File.OpenRead(fileName))
    {
        return (Foo)serializer.Deserialize(stream);
    }
}
```

:

```
<Foo>
    <Dog/>
</Foo>
```

-

```
public class Foo
{
    // Using XmlElement
    [XmlElement(Name="Dog")]
    public Animal Cat { get; set; }
}
```

: (XmlArray)

```
<Store>
    <Articles>
        <Product/>
        <Product/>
    </Articles>
</Store>
```

-

```
public class Store
{
    [XmlArray("Articles")]
    public List<Product> Products {get; set; }
}
```

: DateTime

```
public class Dog
{
    private const string _birthStringFormat = "yyyy-MM-dd";

    [XmlIgnore]
    public DateTime Birth {get; set;}

    [XmlElement(ElementName="Birth")]
    public string BirthString
    {
        get { return Birth.ToString(_birthStringFormat); }
        set { Birth = DateTime.ParseExact(value, _birthStringFormat,
CultureInfo.InvariantCulture); }
    }
}
```

serializer

XmlSerializer . , . . .

?

`new XmlSerializer(type, knownTypes)` N serializer O (N ^ 2) . .

```
// Beware of the N^2 in terms of the number of types.
var allSerializers = allTypes.Select(t => new XmlSerializer(t, allTypes));
var serializerDictionary = Enumerable.Range(0, allTypes.Length)
    .ToDictionary (i => allTypes[i], i => allSerializers[i])
```

OOP .

. serializer .

System.Xml.Serialization.XmlSerializer.FromTypes ([])

FromTypes Type XmlSerializer .

```
var allSerializers = XmlSerializer.FromTypes(allTypes);
var serializerDictionary = Enumerable.Range(0, allTypes.Length)
    .ToDictionary(i => allTypes[i], i => allSerializers[i]);
```

.

```
using System;
using System.Collections.Generic;
using System.Xml.Serialization;
using System.Linq;
using System.Linq;

public class Program
{
    public class Container
    {
        public Base Base { get; set; }
    }

    public class Base
    {
        public int JustSomePropInBase { get; set; }
    }

    public class Derived : Base
    {
        public int JustSomePropInDerived { get; set; }
    }

    public void Main()
    {
        var sampleObject = new Container { Base = new Derived() };
        var allTypes = new[] { typeof(Container), typeof(Base), typeof(Derived) };

        Console.WriteLine("Trying to serialize without a derived class metadata:");
        SetupSerializers(allTypes.Except(new[] { typeof(Derived) }).ToArray());
        try
        {
            Serialize(sampleObject);
        }
        catch (InvalidOperationException e)
        {
            Console.WriteLine();
            Console.WriteLine("This error was anticipated,");
            Console.WriteLine("we have not supplied a derived class.");
            Console.WriteLine(e);
        }
        Console.WriteLine("Now trying to serialize with all of the type information:");
        SetupSerializers(allTypes);
        Serialize(sampleObject);
        Console.WriteLine();
        Console.WriteLine("Slides down well this time!");
    }

    static void Serialize<T>(T o)
    {
        serializerDictionary[typeof(T)].Serialize(Console.Out, o);
    }

    private static Dictionary<Type, XmlSerializer> serializerDictionary;

    static void SetupSerializers(Type[] allTypes)
    {
        var allSerializers = XmlSerializer.FromTypes(allTypes);
        serializerDictionary = Enumerable.Range(0, allTypes.Length)
            .ToDictionary(i => allTypes[i], i => allSerializers[i]);
    }
```

```
}
```

:

```
Trying to serialize without a derived class metadata:
<?xml version="1.0" encoding="utf-16"?>
<Container xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
This error was anticipated,
we have not supplied a derived class.
System.InvalidOperationException: There was an error generating the XML document. --->
System.InvalidOperationException: The type Program+Derived was not expected. Use the
XmlInclude or SoapInclude attribute to specify types that are not known statically.
   at Microsoft.Xml.Serialization.GeneratedAssembly.XmlSerializationWriter1.Write2_Base(String
n, String ns, Base o, Boolean isNullable, Boolean needType)
   at
Microsoft.Xml.Serialization.GeneratedAssembly.XmlSerializationWriter1.Write3_Container(String
n, String ns, Container o, Boolean isNullable, Boolean needType)
   at
Microsoft.Xml.Serialization.GeneratedAssembly.XmlSerializationWriter1.Write4_Container(Object
o)
   at System.Xml.Serialization.XmlSerializer.Serialize(XmlWriter xmlWriter, Object o,
XmlSerializerNamespaces namespaces, String encodingStyle, String id)
   --- End of inner exception stack trace ---
   at System.Xml.Serialization.XmlSerializer.Serialize(XmlWriter xmlWriter, Object o,
XmlSerializerNamespaces namespaces, String encodingStyle, String id)
   at System.Xml.Serialization.XmlSerializer.Serialize(XmlWriter xmlWriter, Object o,
XmlSerializerNamespaces namespaces, String encodingStyle)
   at System.Xml.Serialization.XmlSerializer.Serialize(XmlWriter xmlWriter, Object o,
XmlSerializerNamespaces namespaces)
   at Program.Serialize[T](T o)
   at Program.Main()
Now trying to serialize with all of the type information:
<?xml version="1.0" encoding="utf-16"?>
<Container xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Base xsi:type="Derived">
    <JustSomePropInBase>0</JustSomePropInBase>
    <JustSomePropInDerived>0</JustSomePropInDerived>
  </Base>
</Container>
Slides down well this time!
```

( ) - :

```
Use the XmlInclude or SoapInclude attribute to specify types that are not known statically.
```

XML .

```
<Base xsi:type="Derived">
```

```
Base  Container       Derived .
```

.

XmlSerializer : https://riptutorial.com/ko/dot-net/topic/31/xmlserializer

26: Zip

ZipFile System.IO.Compression . Zip Zip .

- FileStream MemoryStream .
-

ArgumentException	(:).
ArgumentNullException	null
ArgumentOutOfRangeException	.
InvalidDataException	

InvalidDataException (), 3 .

- zip .
- .
- Update .

MSDN .

Examples

ZIP

zip . zip .

```
using (FileStream fs = new FileStream("archive.zip", FileMode.Open))
using (ZipArchive archive = new ZipArchive(fs, ZipArchiveMode.Read))
{
    for (int i = 0; i < archive.Entries.Count; i++)
    {
        Console.WriteLine($"{i}: {archive.Entries[i]}");
    }
}
```

ZIP

.

```
using (FileStream fs = new FileStream("archive.zip", FileMode.Open))
using (ZipArchive archive = new ZipArchive(fs, ZipArchiveMode.Read))
{
    archive.ExtractToDirectory(AppDomain.CurrentDomain.BaseDirectory);
}
```

System.IO.IOException .

:

```
using (FileStream fs = new FileStream("archive.zip", FileMode.Open))
using (ZipArchive archive = new ZipArchive(fs, ZipArchiveMode.Read))
{
    // Get a root entry file
    archive.GetEntry("test.txt").ExtractToFile("test_extracted_getentries.txt", true);

    // Enter a path if you want to extract files from a subdirectory
    archive.GetEntry("sub/subtest.txt").ExtractToFile("test_sub.txt", true);

    // You can also use the Entries property to find files
    archive.Entries.FirstOrDefault(f => f.Name ==
"test.txt")?.ExtractToFile("test_extracted_linq.txt", true);

    // This will throw a System.ArgumentNullException because the file cannot be found
    archive.GetEntry("nonexistingfile.txt").ExtractToFile("fail.txt", true);
}
```

.

ZIP

ZIP ZipArchiveMode.Update .

```
using (FileStream fs = new FileStream("archive.zip", FileMode.Open))
using (ZipArchive archive = new ZipArchive(fs, ZipArchiveMode.Update))
{
    // Add file to root
    archive.CreateEntryFromFile("test.txt", "test.txt");

    // Add file to subfolder
    archive.CreateEntryFromFile("test.txt", "symbols/test.txt");
}
```

:

```
var entry = archive.CreateEntry("createentry.txt");
using(var writer = new StreamWriter(entry.Open()))
{
    writer.WriteLine("Test line");
}
```

Zip : https://riptutorial.com/ko/dot-net/topic/9943/zip----

27:

?

.NET `ForEach` . `foreach` .

Examples

```
public class Customer {
   public void SendEmail()
   {
       // Sending email code here
   }
}

List<Customer> customers = new List<Customer>();

customers.Add(new Customer());
customers.Add(new Customer());

customers.ForEach(c => c.SendEmail());
```

IEnumerable

`ForEach()` `List<T>` `IQueryable<T>` `IEnumerable<T>` . .

.

( ) . .

```
IEnumerable<Customer> customers = new List<Customer>();

customers.ToList().ForEach(c => c.SendEmail());
```

.

.

```
public static void ForEach<T>(this IEnumerable<T> enumeration, Action<T> action)
{
    foreach(T item in enumeration)
    {
        action(item);
    }
}
```

:

```
IEnumerable<Customer> customers = new List<Customer>();

customers.ForEach(c => c.SendEmail());
```

: LINQ . `ForEach` . `foreach` .

: https://riptutorial.com/ko/dot-net/topic/2225/

28:

MEF .

System.ComponentModel.Composition .

() .

```
using System.Collections.ObjectModel;

namespace Demo
{
    public sealed class User
    {
        public User(int id, string name)
        {
            this.Id = id;
            this.Name = name;
        }

        public int Id { get; }
        public string Name { get; }
        public override string ToString() => $"User[Id: {this.Id}, Name={this.Name}]";
    }

    public interface IUserProvider
    {
        ReadOnlyCollection<User> GetAllUsers();
    }
}
```

Examples

()

```
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel.Composition;

namespace Demo
{
    [Export(typeof(IUserProvider))]
    public sealed class UserProvider : IUserProvider
    {
        public ReadOnlyCollection<User> GetAllUsers()
        {
            return new List<User>
            {
                new User(0, "admin"),
                new User(1, "Dennis"),
                new User(2, "Samantha"),
            }.AsReadOnly();
        }
    }
```

```
}
```

. (ComposablePartCatalog) .

()

```
using System;
using System.ComponentModel.Composition;

namespace Demo
{
    public sealed class UserWriter
    {
        [Import(typeof(IUserProvider))]
        private IUserProvider userProvider;

        public void PrintAllUsers()
        {
            foreach (User user in this.userProvider.GetAllUsers())
            {
                Console.WriteLine(user);
            }
        }
    }
}
```

`IUserProvider` . (ComposablePartCatalog) .

()

() .

```
using System.ComponentModel.Composition;
using System.ComponentModel.Composition.Hosting;

namespace Demo
{
    public static class Program
    {
        public static void Main()
        {
            using (var catalog = new ApplicationCatalog())
            using (var exportProvider = new CatalogExportProvider(catalog))
            using (var container = new CompositionContainer(exportProvider))
            {
                exportProvider.SourceProvider = container;

                UserWriter writer = new UserWriter();

                // at this point, writer's userProvider field is null
                container.ComposeParts(writer);

                // now, it should be non-null (or an exception will be thrown).
                writer.PrintAllUsers();
            }
        }
```

```
    }
}
```

`[Export(typeof(IUserProvider))]` `UserWriter` .

(: `DirectoryCatalog`) `ApplicationCatalog` ( ) .

: https://riptutorial.com/ko/dot-net/topic/62/----

29:

: HTTP

Examples

TCP (TcpListener, TcpClient, NetworkStream)

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Text;

class TcpChat
{
    static void Main(string[] args)
    {
        if(args.Length == 0)
        {
            Console.WriteLine("Basic TCP chat");
            Console.WriteLine();
            Console.WriteLine("Usage:");
            Console.WriteLine("tcpchat server <port>");
            Console.WriteLine("tcpchat client <url> <port>");
            return;
        }

        try
        {
            Run(args);
        }
        catch(IOException)
        {
            Console.WriteLine("--- Connection lost");
        }
        catch(SocketException ex)
        {
            Console.WriteLine("--- Can't connect: " + ex.Message);
        }
    }

    static void Run(string[] args)
    {
        TcpClient client;
        NetworkStream stream;
        byte[] buffer = new byte[256];
        var encoding = Encoding.ASCII;

        if(args[0].StartsWith("s", StringComparison.InvariantCultureIgnoreCase))
        {
            var port = int.Parse(args[1]);
            var listener = new TcpListener(IPAddress.Any, port);
            listener.Start();
            Console.WriteLine("--- Waiting for a connection...");
            client = listener.AcceptTcpClient();
```

```
        }
        else
        {
            var hostName = args[1];
            var port = int.Parse(args[2]);
            client = new TcpClient();
            client.Connect(hostName, port);
        }

        stream = client.GetStream();
        Console.WriteLine("--- Connected. Start typing! (exit with Ctrl-C)");

        while(true)
        {
            if(Console.KeyAvailable)
            {
                var lineToSend = Console.ReadLine();
                var bytesToSend = encoding.GetBytes(lineToSend + "\r\n");
                stream.Write(bytesToSend, 0, bytesToSend.Length);
                stream.Flush();
            }

            if (stream.DataAvailable)
            {
                var receivedBytesCount = stream.Read(buffer, 0, buffer.Length);
                var receivedString = encoding.GetString(buffer, 0, receivedBytesCount);
                Console.Write(receivedString);
            }
        }
    }
}
```

SNTP (UdpClient)

SNTP RFC 2030 .

```
using System;
using System.Globalization;
using System.Linq;
using System.Net;
using System.Net.Sockets;

class SntpClient
{
    const int SntpPort = 123;
    static DateTime BaseDate = new DateTime(1900, 1, 1);

    static void Main(string[] args)
    {
        if(args.Length == 0) {
            Console.WriteLine("Simple SNTP client");
            Console.WriteLine();
            Console.WriteLine("Usage: sntpclient <sntp server url> [<local timezone>]");
            Console.WriteLine();
            Console.WriteLine("<local timezone>: a number between -12 and 12 as hours from
UTC");
            Console.WriteLine("(append .5 for an extra half an hour)");
            return;
        }
```

```
        double localTimeZoneInHours = 0;
        if(args.Length > 1)
            localTimeZoneInHours = double.Parse(args[1], CultureInfo.InvariantCulture);

        var udpClient = new UdpClient();
        udpClient.Client.ReceiveTimeout = 5000;

        var sntpRequest = new byte[48];
        sntpRequest[0] = 0x23; //LI=0 (no warning), VN=4, Mode=3 (client)

        udpClient.Send(
            dgram: sntpRequest,
            bytes: sntpRequest.Length,
            hostname: args[0],
            port: SntpPort);

        byte[] sntpResponse;
        try
        {
            IPEndPoint remoteEndpoint = null;
            sntpResponse = udpClient.Receive(ref remoteEndpoint);
        }
        catch(SocketException)
        {
            Console.WriteLine("*** No response received from the server");
            return;
        }

        uint numberOfSeconds;
        if(BitConverter.IsLittleEndian)
            numberOfSeconds = BitConverter.ToUInt32(
                sntpResponse.Skip(40).Take(4).Reverse().ToArray()
                ,0);
        else
            numberOfSeconds = BitConverter.ToUInt32(sntpResponse, 40);

        var date = BaseDate.AddSeconds(numberOfSeconds).AddHours(localTimeZoneInHours);

        Console.WriteLine(
            $"Current date in server: {date:yyyy-MM-dd HH:mm:ss}
UTC{localTimeZoneInHours:+0.#;-0.#;.}");
    }
}
```

: https://riptutorial.com/ko/dot-net/topic/35/

30:

Examples

MSTest

- .
- .
- - `Foo` `Foo.Tests`
- .

MSTest ( ) `[TestClass]` `[TestMethod]` `[TestMethod]` .

```
[TestClass]
public class FizzBuzzFixture
{
    [TestMethod]
    public void Test1()
    {
        //arrange
        var solver = new FizzBuzzSolver();
        //act
        var result = solver.FizzBuzz(1);
        //assert
        Assert.AreEqual("1",result);
    }
}
```

: https://riptutorial.com/ko/dot-net/topic/5365/-

31:

.NET . . .

Examples

. (,) . .NET Parallel.For Parallel.Foreach .

```
//Sequential version

foreach (var item in sourcecollection){

Process(item);

}

// Parallel equivalent

Parallel.foreach(sourcecollection, item => Process(item));
```

Parallel.ForEach .

: https://riptutorial.com/ko/dot-net/topic/8085/-----

32:

(Synchronization Context) .

. WPF Winforms UI `SynchronizationContext` . `SynchronizationContext` - . () UI .

.

Examples

UI

`SynchronizationContext` UI .

```
void Button_Click(object sender, EventArgs args)
{
    SynchronizationContext context = SynchronizationContext.Current;
    Task.Run(() =>
    {
        for(int i = 0; i < 10; i++)
        {
            Thread.Sleep(500); //simulate work being done
            context.Post(ShowProgress, "Work complete on item " + i);
        }
    }
}

void UpdateCallback(object state)
{
    // UI can be safely updated as this method is only called from the UI thread
    this.MyTextBox.Text = state as string;
}
```

`for MyTextBox.Text .UpdateCallback SynchronizationContext` UI .

`System.IProgress<T> . System.Progress<T>` .

: https://riptutorial.com/ko/dot-net/topic/5407/-

33:

.NET GC . GC . GC .

Examples

GC "heap" *(heap)* . (:). . .NET `IDisposable` . `Dispose` .

```
public interface IDisposable
{
    Dispose();
}
```

. `finally` `using` `Dispose()` .

```
StreamReader sr;
string textFromFile;
string filename = "SomeFile.txt";
try
{
    sr = new StreamReader(filename);
    textFromFile = sr.ReadToEnd();
}
finally
{
    if (sr != null) sr.Dispose();
}
```

```
string textFromFile;
string filename = "SomeFile.txt";

using (StreamReader sr = new Streamreader(filename))
{
    textFromFile = sr.ReadToEnd();
}
```

.

SafeHandle

`IDisposable` finalizer `SafeHandle` . `SafeHandle` . SafeHandle API . `SafeHandle` .

```
using System.Runtime.InteropServices;

class MyHandle : SafeHandle
{
    public override bool IsInvalid => handle == IntPtr.Zero;
    public MyHandle() : base(IntPtr.Zero, true)
    { }
```

```
    public MyHandle(int length) : this()
    {
        SetHandle(Marshal.AllocHGlobal(length));
    }

    protected override bool ReleaseHandle()
    {
        Marshal.FreeHGlobal(handle);
        return true;
    }
}
```

: `IDisposable` `SafeHandle` Finalizer . .

: https://riptutorial.com/ko/dot-net/topic/59/-

34:

.NET `System.String` `System.Char` UTF-16 . .NET ( ) .

grapheme . . `System.Char` . .

- . à : *U + 0061 A U + 0300* . `"à ".Length == 2` . 1 .
- A *U + 00E0 LATIN SMALL LETTER A* , . `"\u00e0" == "\u0061\u0300"` (`"\u00e0".Length != "\u0061\u0300".Length`) . `String.Normalize()` .
- . , *U + D55C HAN CHARACTER* (UTF-16) , be . .
- : *U + 2008A HAN* `System.Char` (`"\ud840\udc8a"` UTF-16 :) ! ( ) .
- trigraph, *fyzika* .

(?). .

( ).

```
public static class StringExtensions
{
    public static IEnumerable<string> EnumerateCharacters(this string s)
    {
        if (s == null)
            return Enumerable.Empty<string>();

        var enumerator = StringInfo.GetTextElementEnumerator(s.Normalize());
        while (enumerator.MoveNext())
            yield return (string)enumerator.Value;
    }
}
```

Examples

Remarks `Length` `System.Char` Unicode `Length` graphemes). , `text.Distinct().Count()` `text.Distinct().Count()` .

```
int distinctCharactersCount = text.EnumerateCharacters().Count();
```

. ( ).

```
var frequencies = text.EnumerateCharacters()
    .GroupBy(x => x, StringComparer.CurrentCultureIgnoreCase)
    .Select(x => new { Character = x.Key, Count = x.Count() };
```

(`System.Char` Length . graphemes). .

```
int length = text.EnumerateCharacters().Count();
```

`EnumerateCharacters()` .

```
public static class StringExtensions
{
    public static int CountCharacters(this string text)
    {
        if (String.IsNullOrEmpty(text))
            return 0;

        int count = 0;
        var enumerator = StringInfo.GetTextElementEnumerator(text);
        while (enumerator.MoveNext())
            ++count;

        return count;
    }
}
```

.

```
int count = text.Count(x => x == ch);
```

.

```
public static int CountOccurrencesOf(this string text, string character)
{
    return text.EnumerateCharacters()
        .Count(x => String.Equals(x, character, StringComparer.CurrentCulture));
}
```

( ) .

. System.Char System.Char . .(.):

```
public static IEnumerable<string> Split(this string value, int desiredLength)
{
    var characters = StringInfo.GetTextElementEnumerator(value);
    while (characters.MoveNext())
        yield return String.Concat(Take(characters, desiredLength));
}

private static IEnumerable<string> Take(TextElementEnumerator enumerator, int count)
{
    for (int i = 0; i < count; ++i)
    {
        yield return (string)enumerator.Current;

        if (!enumerator.MoveNext())
            yield break;
    }
}
```

.NET System.Char (UTF-16) . ( ) System.Byte .

System.Text.Encoder System.Text.Decoder . System.Text.Encoder (*X* byte[] UTF-16 System.String vice -versa).

/ `System.Text.Encoding` (UTF-8, UTF-16) .

:

UTF-8 .

```
byte[] data = Encoding.UTF8.GetBytes("This is my text");
```

UTF-8

```
var text = Encoding.UTF8.GetString(data);
```

UTF-8 UTF-16 . .

```
var content = File.ReadAllText(path, Encoding.UTF8);
File.WriteAllText(content, Encoding.UTF16);
```

Object.ToString ()

.NET `Object` `ToString()` . . , .

```
public class Foo
{
}

var foo = new Foo();
Console.WriteLine(foo); // outputs Foo
```

`ToString()` concatinating .

```
public class Foo
{
    public override string ToString()
    {
        return "I am Foo";
    }
}

var foo = new Foo();
Console.WriteLine("I am bar and "+foo);// outputs I am bar and I am Foo
```

. DebuggerDisplay Attribute (MSDN) .

```
// [DebuggerDisplay("Person = FN {FirstName}, LN {LastName}")]
[DebuggerDisplay("Person = FN {"+nameof(Person.FirstName)+"}, LN
{"+nameof(Person.LastName)+"}")]
```

```
public class Person
{
    public string FirstName { get; set; }
    public string LastName { get; set;}
    // ...
}
```

. . . .

```
string veryLongString = ...
// memory is allocated
string newString = veryLongString.Remove(0,1); // removes first character of the string.
```

StringBuilder .

```
var sb = new StringBuilder(someInitialString);
foreach(var str in manyManyStrings)
{
    sb.Append(str);
}
var finalString = sb.ToString();
```

Comparing strings

String == .

string . . ( ). .

Length !

StringComparison String.Equals .

: https://riptutorial.com/ko/dot-net/topic/2227/

35:

Examples

?

CLR (Common Language Runtime) . , .

```
using System.Reflection;
```

```
Assembly assembly = this.GetType().Assembly;
```

, IL .

```
Assembly assembly = Assembly.GetExecutingAssembly();

foreach (var type in assembly.GetTypes())
{
    Console.WriteLine(type.FullName);
}
```

ID .

```
Console.WriteLine(typeof(int).Assembly.FullName);
// Will print: "mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
```

PublicKeyToken . . PublicKeyToken . . .

Reflection T

```
T variable = Activator.CreateInstance(typeof(T));
```

```
T variable = Activator.CreateInstance(typeof(T), arg1, arg2);
```

property¦ Classy .

```
public class Classy
{
    public string Propertua {get; set;}
}
```

Propertua Propertua Propertua :

```
var typeOfClassy = typeof (Classy);
var classy = new Classy();
var prop = typeOfClassy.GetProperty("Propertua");
prop.SetValue(classy, "Value");
```

( ) enum

. .

```
    private static Dictionary<object, object> attributeCache = new Dictionary<object,
object>();

    public static T GetAttribute<T, V>(this V value)
        where T : Attribute
        where V : struct
    {
        object temp;

        // Try to get the value from the static cache.
        if (attributeCache.TryGetValue(value, out temp))
        {
            return (T) temp;
        }
        else
        {
            // Get the type of the struct passed in.
            Type type = value.GetType();
            FieldInfo fieldInfo = type.GetField(value.ToString());

            // Get the custom attributes of the type desired found on the struct.
            T[] attribs = (T[])fieldInfo.GetCustomAttributes(typeof(T), false);

            // Return the first if there was a match.
            var result = attribs.Length > 0 ? attribs[0] : null;

            // Cache the result so future checks won't need reflection.
            attributeCache.Add(value, result);

            return result;
        }
    }
```

```
public class Equatable
{
    public string field1;

    public override bool Equals(object obj)
    {
        if (ReferenceEquals(null, obj)) return false;
        if (ReferenceEquals(this, obj)) return true;

        var type = obj.GetType();
        if (GetType() != type)
            return false;

        var fields = type.GetFields(BindingFlags.Instance | BindingFlags.NonPublic |
BindingFlags.Public);
        foreach (var field in fields)
            if (field.GetValue(this) != field.GetValue(obj))
                return false;

        return true;
    }
```

```
    public override int GetHashCode()
    {
        var accumulator = 0;
        var fields = GetType().GetFields(BindingFlags.Instance | BindingFlags.NonPublic |
BindingFlags.Public);
        foreach (var field in fields)
            accumulator = unchecked ((accumulator * 937) ^
field.GetValue(this).GetHashCode());

        return accumulator;
    }
}
```

: comparasion ( ) .

: https://riptutorial.com/ko/dot-net/topic/44/

36:

`struct` . . (Microsoft `struct` 16).

`class` C # ( ) .

`enum` . . `constant` Intellisense .

Examples

Structs System.ValueType . .

```
Struct MyStruct
{
    public int x;
    public int y;
}
```

. `AddNumbers` Value `int a b` .

```
int a = 5;
int b = 6;

AddNumbers(a,b);

public AddNumbers(int x, int y)
{
    int z = x + y; // z becomes 11
    x = x + 5; // now we changed x to be 10
    z = x + y; // now z becomes 16
}
```

5 `x a , , x a a .`

.

System.Object . .

```
public Class MyClass
{
    public int a;
    public int b;
}
```

, . . `int` "".

```
MyClass instanceOfMyClass = new MyClass();
instanceOfMyClass.a = 5;
instanceOfMyClass.b = 6;
```

```
AddNumbers(instanceOfMyClass);

public AddNumbers(MyClass sample)
{
    int z = sample.a + sample.b; // z becomes 11
    sample.a = sample.a + 5; // now we changed a to be 10
    z = sample.a + sample.b; // now z becomes 16
}
```

`sample.a` 10 `instanceOfMyClass.a` . ( ) .

.

Enum

. `enum` **System.Enum** . .

```
public enum MyEnum
{
    Monday = 1,
    Tuesday,
    Wednesday,
    //...
}
```

. enum `1` . `Tuesday` `2` , `Wednesday` `3` .

enum `int` 0 `byte, sbyte, short, ushort, int, uint, long, or ulong` .. , 1 .

MyEnum .

```
MyEnum instance = (MyEnum)3; // the variable named 'instance' gets a
                             //value of MyEnum.Wednesday, which maps to 3.

int x = 2;
instance = (MyEnum)x; // now 'instance' has a value of MyEnum.Tuesday
```

enum `Flags` . `Flags` . 2 .

```
[Flags]
public enum MyEnum
{
    Monday = 1,
    Tuesday = 2,
    Wednesday = 4,
    Thursday = 8,
    Friday = 16,
    Saturday = 32,
    Sunday = 64
}
```

.NET 4.0 `Enum.HasFlag` .

```
MyEnum instance = MyEnum.Monday | MyEnum.Thursday; // instance now has a value of
                                                   // *both* Monday and Thursday,
                                                   // represented by (in binary) 0100.

if (instance.HasFlag(MyEnum.Wednesday))
{
    // it doesn't, so this block is skipped
}
else if (instance.HasFlag(MyEnum.Thursday))
{
    // it does, so this block is executed
}
```

Enum `System.ValueType` . enum Enum (System.Int32) . . ValueType () (MSDN) .

: https://riptutorial.com/ko/dot-net/topic/57/--

37:

Examples

.

KeyValue

```
Dictionary<int, string> dict = new Dictionary<int, string>();
foreach(KeyValuePair<int, string> kvp in dict)
{
   Console.WriteLine("Key : " + kvp.Key.ToString() + ", Value : " + kvp.Value);
}
```

```
Dictionary<int, string> dict = new Dictionary<int, string>();
foreach(int key in dict.Keys)
{
    Console.WriteLine("Key : " + key.ToString() + ", Value : " + dict[key]);
}
```

```
Dictionary<int, string> dict = new Dictionary<int, string>();
foreach(string s in dict.Values)
{
    Console.WriteLine("Value : " + s);
}
```

```
// Translates to `dict.Add(1, "First")` etc.
var dict = new Dictionary<int, string>()
{
    { 1, "First" },
    { 2, "Second" },
    { 3, "Third" }
};

// Translates to `dict[1] = "First"` etc.
// Works in C# 6.0.
var dict = new Dictionary<int, string>()
{
    [1] = "First",
    [2] = "Second",
    [3] = "Third"
};
```

```
Dictionary<int, string> dict = new Dictionary<int, string>();
dict.Add(1, "First");
dict.Add(2, "Second");

// To safely add items (check to ensure item does not already exist - would throw)
if(!dict.ContainsKey(3))
{
   dict.Add(3, "Third");
}
```

/ . get set .

```
Dictionary<int, string> dict = new Dictionary<int, string>();
dict[1] = "First";
dict[2] = "Second";
dict[3] = "Third";
```

throw `Add` .

`ConcurrentDictionary<TKey, TValue>` :

```
var dict = new ConcurrentDictionary<int, string>();
dict.AddOrUpdate(1, "First", (oldKey, oldValue) => "First");
```

:

```
var dict = new Dictionary<int, string>()
{
    { 1, "First" },
    { 2, "Second" },
    { 3, "Third" }
};
```

1 . `KeyNotFoundException` `ContainsKey` `ContainsKey` .

```
if (dict.ContainsKey(1))
    Console.WriteLine(dict[1]);
```

. ( ). . .

```
string value;
if (dict.TryGetValue(1, out value))
    Console.WriteLine(value);
```

Case-Insensivitve .

```
var MyDict = new Dictionary<string,T>(StringComparison.InvariantCultureIgnoreCase)
```

(.NET 4.0)

/ .

`Dictionary<TKey, TValue>` . :

```
var dict = new ConcurrentDictionary<int, string>();
```

`Add` . `AddOrUpdate` .

`AddOrUpdate(TKey key, TValue, Func<TKey, TValue, TValue> addValue)` - / / . .

(2) `AddOrUpdate(TKey key, Func<TKey, TValue> addValue, Func<TKey, TValue, TValue> updateValueFactory)` - / . / .

key (1) .

```
string addedValue = dict.AddOrUpdate(1, "First", (updateKey, valueOld) => "First");
```

(1) .

```
string addedValue2 = dict.AddOrUpdate(1, "First", (updateKey, valueOld) => $"{valueOld}
Updated");
```

overload (2) .

```
string addedValue3 = dict.AddOrUpdate(1, (key) => key == 1 ? "First" : "Not First",
(updateKey, valueOld) => $"{valueOld} Updated");
```

`Dictionary<TKey,TValue>` .

```
string value = null;
bool success = dict.TryGetValue(1, out value);
```

(thread-safe) .

2 , " " .

```
string theValue = dict.GetOrAdd(2, "Second");
```

:

```
string theValue2 = dict.GetOrAdd(2, (key) => key == 2 ? "Second" : "Not Second." );
```

IEnumerable (≥ .NET 3.5)

IEnumerable <T> <TKey, TValue> :

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
public class Fruits
{
    public int Id { get; set; }
    public string Name { get; set; }
}
```

```
var fruits = new[]
{
```

```
    new Fruits { Id = 8 , Name = "Apple" },
    new Fruits { Id = 3 , Name = "Banana" },
    new Fruits { Id = 7 , Name = "Mango" },
};


// Dictionary<int, string>                  key      value
var dictionary = fruits.ToDictionary(x => x.Id, x => x.Name);
```

:

```
var dict = new Dictionary<int, string>()
{
    { 1, "First" },
    { 2, "Second" },
    { 3, "Third" }
};
```

Remove .

```
bool wasRemoved = dict.Remove(2);
```

2 . Remove . false ( ).

null .

```
dict[2] = null; // WRONG WAY TO REMOVE!
```

. null null .

Clear .

```
dict.Clear();
```

Clear Count 0 .

ContainsKey (TKey)

Dictionary ContainsKey(TKey) TKey . bool . :

```
var dictionary = new Dictionary<string, Customer>()
{
   {"F1", new Customer() { FirstName = "Felipe", ... } },
   {"C2", new Customer() { FirstName = "Carl", ... } },
   {"J7", new Customer() { FirstName = "John", ... } },
   {"M5", new Customer() { FirstName = "Mary", ... } },
};
```

C2 .

```
if (dictionary.ContainsKey("C2"))
```

```
{
   // exists
}
```

ContainsKey `Dictionary<TKey, TValue>` .

KeyValuePair :

```
Dictionary<int, int> dictionary = new Dictionary<int, int>();
List<KeyValuePair<int, int>> list = new List<KeyValuePair<int, int>>();
list.AddRange(dictionary);
```

:

```
Dictionary<int, int> dictionary = new Dictionary<int, int>();
List<int> list = new List<int>();
list.AddRange(dictionary.Keys);
```

:

```
Dictionary<int, int> dictionary = new Dictionary<int, int>();
List<int> list = new List<int>();
list.AddRange(dictionary.Values);
```

ConcurrentDictionary Lazy'1 .

ConcurrentDictionary . ?

CPU . . .

`ConcurrentDictionary<TKey, TValue> Lazy<TValue>` . ConcurrentDictionary GetOrAdd . Lazy Lazy . Lazy Value GetOrAdd Value .

```
public static class ConcurrentDictionaryExtensions
{
    public static TValue GetOrCreateLazy<TKey, TValue>(
        this ConcurrentDictionary<TKey, Lazy<TValue>> d,
        TKey key,
        Func<TKey, TValue> factory)
    {
        return
            d.GetOrAdd(
                key,
                key1 =>
                    new Lazy<TValue>(() => factory(key1),
                    LazyThreadSafetyMode.ExecutionAndPublication)).Value;
    }
}
```

XmlSerializer . : , . ConcurrentDictionary . Lazy ConcurrentDictionary .

```
private ConcurrentDictionary<Type, Lazy<XmlSerializer>> _serializers =
    new ConcurrentDictionary<Type, Lazy<XmlSerializer>>();

public XmlSerializer GetSerialier(Type t)
{
    return _serializers.GetOrCreateLazy(t, BuildSerializer);
}

private XmlSerializer BuildSerializer(Type t)
{
    throw new NotImplementedException("and this is a homework");
}
```

: https://riptutorial.com/ko/dot-net/topic/45/

38:

Examples

.NET 1.x ConfigurationSettings AppSettings

ConfigurationSettings .NET 1.0 1.1 . ConfigurationManager WebConfigurationManager .

`appSettings` .

app.config

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <appSettings>
    <add key="keyName" value="anything, as a string"/>
    <add key="keyNames" value="123"/>
    <add key="keyNames" value="234"/>
  </appSettings>
</configuration>
```

Program.cs

```
using System;
using System.Configuration;
using System.Diagnostics;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main()
        {
            string keyValue = ConfigurationSettings.AppSettings["keyName"];
            Debug.Assert("anything, as a string".Equals(keyValue));

            string twoKeys = ConfigurationSettings.AppSettings["keyNames"];
            Debug.Assert("234".Equals(twoKeys));

            Console.ReadKey();
        }
    }
}
```

.NET 2.0 ConfigurationManager AppSettings

ConfigurationManager `AppSettings` . .NET 1.x `appSettings` .

app.config

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
```

```
  <appSettings>
    <add key="keyName" value="anything, as a string"/>
    <add key="keyNames" value="123"/>
    <add key="keyNames" value="234"/>
  </appSettings>
</configuration>
```

Program.cs

```
using System;
using System.Configuration;
using System.Diagnostics;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main()
        {
            string keyValue = ConfigurationManager.AppSettings["keyName"];
            Debug.Assert("anything, as a string".Equals(keyValue));

            var twoKeys = ConfigurationManager.AppSettings["keyNames"];
            Debug.Assert("234".Equals(twoKeys));

            Console.ReadKey();
        }
    }
}
```

Visual Studio

Visual Studio . `appSettings` .

1. . .
2. . `web.config` , .exe.config app.config . *assembly* . ( ) Application Data ( ) `user.config` .
3. .

. . .

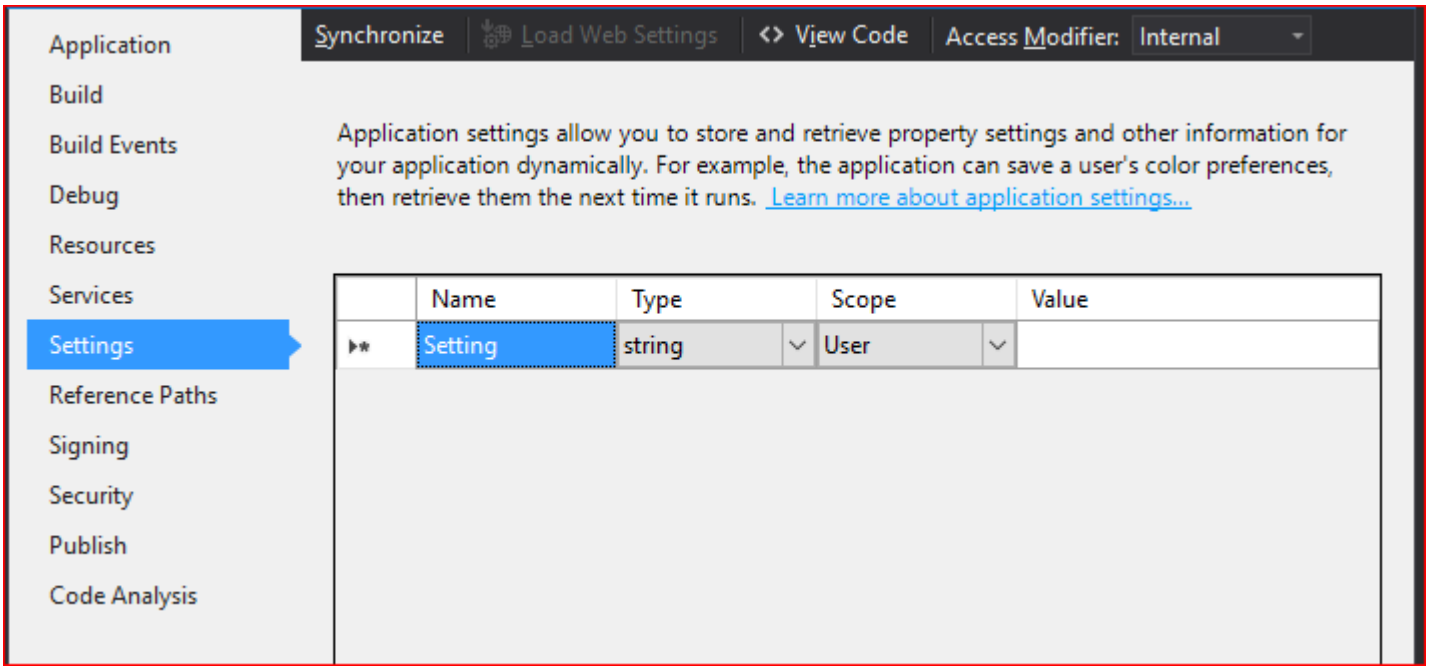
1. (`app.config` `web.config`) .
2. , .
3. `Settings.settings` . .
4. `Properties` . `Settings.Designer.__` (.cs, .vb) `Settings` . . Singleton Pattern Singleton `Default` .

Visual Studio .

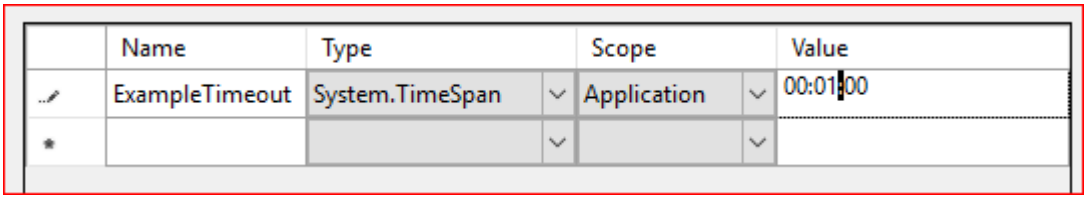
1. .

2. Settings Settings .

Settings :



ExampleTimeout System.Timespan 1 .



Settings .

(C #) .

Program.cs

```
using System;
using System.Diagnostics;
using ConsoleApplication1.Properties;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main()
        {
            TimeSpan exampleTimeout = Settings.Default.ExampleTimeout;
            Debug.Assert(TimeSpan.FromMinutes(1).Equals(exampleTimeout));

            Console.ReadKey();
        }
    }
```

```
}
```

:

app.config (Visual Studio)

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <sectionGroup name="applicationSettings"
type="System.Configuration.ApplicationSettingsGroup, System, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089" >
      <section name="ConsoleApplication1.Properties.Settings"
type="System.Configuration.ClientSettingsSection, System, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089" requirePermission="false" />
    </sectionGroup>
  </configSections>
  <appSettings />
  <applicationSettings>
    <ConsoleApplication1.Properties.Settings>
      <setting name="ExampleTimeout" serializeAs="String">
        <value>00:01:00</value>
      </setting>
    </ConsoleApplication1.Properties.Settings>
  </applicationSettings>
</configuration>
```

appSettings . applicationSettings setting . . Settings .

Settings ConfigurationManager .

Settings.designer.cs (C #)

```
...
    [global::System.Configuration.ApplicationScopedSettingAttribute()]
    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.Configuration.DefaultSettingValueAttribute("00:01:00")]
    public global::System.TimeSpan ExampleTimeout {
        get {
            return ((global::System.TimeSpan)(this["ExampleTimeout"]));
        }
    }
...
```

DefaultSettingValueAttribute . .

: https://riptutorial.com/ko/dot-net/topic/54/

39:

Examples

GUI . .

" : 'control_name' () ."

system.windows.forms .

```
private void button4_Click(object sender, EventArgs e)
{
    Thread thread = new Thread(updatetextbox);
    thread.Start();
}

private void updatetextbox()
{
    textBox1.Text = "updated"; // Throws exception
}
```

Control.Invoke Control.BeginInvoke . Control.InvokeRequired .

```
private void updatetextbox()
{
    if (textBox1.InvokeRequired)
        textBox1.BeginInvoke((Action)(() => textBox1.Text = "updated"));
    else
        textBox1.Text = "updated";
}
```

.

```
public static class Extensions
{
    public static void BeginInvokeIfRequired(this ISynchronizeInvoke obj, Action action)
    {
        if (obj.InvokeRequired)
            obj.BeginInvoke(action, new object[0]);
        else
            action();
    }
}
```

.

```
private void updatetextbox()
{
    textBox1.BeginInvokeIfRequired(() => textBox1.Text = "updated");
}
```

Control.BeginInvoke Control.BeginInvoke .

textBox1 Control.Invoke . GUI .

: https://riptutorial.com/ko/dot-net/topic/3098/

40:

`null` . .
Nullable .

Examples

.

System.ValueType .

.
. int 32 . .

. (System.ValueType).

. , .

```
struct PersonAsValueType
{
    public string Name;
}

class Program
{
    static void Main()
    {
        PersonAsValueType personA;

        personA.Name = "Bob";

        var personB = personA;

        personA.Name = "Linda";

        Console.WriteLine(                // Outputs 'False' - because
            object.ReferenceEquals(       // personA and personB are referencing
                personA,                  // different areas of memory
                personB));

        Console.WriteLine(personA.Name);  // Outputs 'Linda'
        Console.WriteLine(personB.Name);  // Outputs 'Bob'
    }
}
```

.
C / C ++ .

(heap) .
. . .

.NET CLR .

. .

. .

.
.

```
class PersonAsReferenceType
{
    public string Name;
}

class Program
{
    static void Main()
    {
        PersonAsReferenceType personA;

        personA = new PersonAsReferenceType { Name = "Bob" };

        var personB = personA;

        personA.Name = "Linda";

        Console.WriteLine(               // Outputs 'True' - because
            object.ReferenceEquals(      // personA and personB are referencing
                personA,                 // the *same* memory location
                personB));

        Console.WriteLine(personA.Name); // Outputs 'Linda'
        Console.WriteLine(personB.Name); // Outputs 'Linda'
    }
```

: https://riptutorial.com/ko/dot-net/topic/9358/--

41: SpeechRecognitionEngine

- SpeechRecognitionEngine ()
- SpeechRecognitionEngine.LoadGrammar ( )
- SpeechRecognitionEngine.SetInputToDefaultAudioDevice ()
- SpeechRecognitionEngine.RecognizeAsync (RecognizeMode)
- GrammarBuilder ()
- GrammarBuilder.Append ( )
- (params string [])
- (GrammarBuilder builder)

LoadGrammar :	
	. , DictationGrammar .
RecognizeAsync :	
	RecognizeMode RecognizeMode : Single RecognizeMode Single , Multiple .
GrammarBuilder.Append :	
	. , "" .
Choices :	
	. GrammarBuilder.Append .
Grammar :	
	GrammarBuilder Grammar .

SpeechRecognitionEngine Windows .

System.Speech.dll .

Examples

```
using System.Speech.Recognition;

// ...

SpeechRecognitionEngine recognitionEngine = new SpeechRecognitionEngine();
recognitionEngine.LoadGrammar(new DictationGrammar());
recognitionEngine.SpeechRecognized += delegate(object sender, SpeechRecognizedEventArgs e)
{
    Console.WriteLine("You said: {0}", e.Result.Text);
};
```

```
recognitionEngine.SetInputToDefaultAudioDevice();
recognitionEngine.RecognizeAsync(RecognizeMode.Multiple);
```

```
SpeechRecognitionEngine recognitionEngine = new SpeechRecognitionEngine();
GrammarBuilder builder = new GrammarBuilder();
builder.Append(new Choices("I am", "You are", "He is", "She is", "We are", "They are"));
builder.Append(new Choices("friendly", "unfriendly"));
recognitionEngine.LoadGrammar(new Grammar(builder));
recognitionEngine.SpeechRecognized += delegate(object sender, SpeechRecognizedEventArgs e)
{
    Console.WriteLine("You said: {0}", e.Result.Text);
};
recognitionEngine.SetInputToDefaultAudioDevice();
recognitionEngine.RecognizeAsync(RecognizeMode.Multiple);
```

SpeechRecognitionEngine : https://riptutorial.com/ko/dot-net/topic/69/--speechrecognitionengine-

42:

Examples

Stopwatch .

```
using System;
using System.Diagnostics;

public class Benchmark : IDisposable
{
    private Stopwatch sw;

    public Benchmark()
    {
        sw = Stopwatch.StartNew();
    }

    public void Dispose()
    {
        sw.Stop();
        Console.WriteLine(sw.Elapsed);
    }
}

public class Program
{
    public static void Main()
    {
        using (var bench = new Benchmark())
        {
            Console.WriteLine("Hello World");
        }
    }
}
```

```
string strCmdText = "/C copy /b Image1.jpg + Archive.rar Image2.jpg";
System.Diagnostics.Process.Start("CMD.exe",strCmdText);
```

cmd .

```
System.Diagnostics.Process process = new System.Diagnostics.Process();
System.Diagnostics.ProcessStartInfo startInfo = new System.Diagnostics.ProcessStartInfo();
startInfo.WindowStyle = System.Diagnostics.ProcessWindowStyle.Hidden;
startInfo.FileName = "cmd.exe";
startInfo.Arguments = "/C copy /b Image1.jpg + Archive.rar Image2.jpg";
process.StartInfo = startInfo;
process.Start();
```

CMD

command Cmd.exe ( ) .

```
private static string SendCommand(string command)
{
    var cmdOut = string.Empty;

    var startInfo = new ProcessStartInfo("cmd", command)
    {
        WorkingDirectory = @"C:\Windows\System32", // Directory to make the call from
        WindowStyle = ProcessWindowStyle.Hidden,   // Hide the window
        UseShellExecute = false,                   // Do not use the OS shell to start the
process
        CreateNoWindow = true,                     // Start the process in a new window
        RedirectStandardOutput = true,             // This is required to get STDOUT
        RedirectStandardError = true               // This is required to get STDERR
    };

    var p = new Process {StartInfo = startInfo};

    p.Start();

    p.OutputDataReceived += (x, y) => cmdOut += y.Data;
    p.ErrorDataReceived += (x, y) => cmdOut += y.Data;
    p.BeginOutputReadLine();
    p.BeginErrorReadLine();
    p.WaitForExit();
    return cmdOut;
}
```

```
var servername = "SVR-01.domain.co.za";
var currentUsers = SendCommand($"/C QUERY USER /SERVER:{servername}")
```

currentUsers = " ID IDLE Joe.Bloggs ica-cgp # 0 2 24692 + 13 : 29 25/07/2016 07:50 Jim.McFlannegan ica-cgp # 1 3 . 2016 08:33 Andy.McAnderson ica-cgp # 2 4 . 25/07/2016 08:54 John.Smith ica-cgp # 4 5 14 25/07/2016 08:57 Bob.Bobbington ica-cgp # 5 6 Active 24692 + 13 : 29 25/07/2016 09:05 Tim.Tom ica-cgp # 6 7 . 25/07/2016 09:08 Bob.Joges ica-cgp # 7 8 24692 + 13 : 29 25 / 07 / 2016 09:13 "

. .

```
private static string SendCommand(string command)
{
    var cmdOut = string.Empty;

    var startInfo = new ProcessStartInfo("cmd", command)
    {
        WorkingDirectory = @"C:\Windows\System32",
        WindowStyle = ProcessWindowStyle.Hidden,    // This does not actually work in
conjunction with "runas" - the console window will still appear!
        UseShellExecute = false,
        CreateNoWindow = true,
        RedirectStandardOutput = true,
        RedirectStandardError = true,

        Verb = "runas",
        Domain = "doman1.co.za",
        UserName = "administrator",
        Password = GetPassword()
```

```
    };

    var p = new Process {StartInfo = startInfo};

    p.Start();

    p.OutputDataReceived += (x, y) => cmdOut += y.Data;
    p.ErrorDataReceived += (x, y) => cmdOut += y.Data;
    p.BeginOutputReadLine();
    p.BeginErrorReadLine();
    p.WaitForExit();
    return cmdOut;
}
```

:

```
static SecureString GetPassword()
{
    var plainText = "password123";
    var ss = new SecureString();
    foreach (char c in plainText)
    {
        ss.AppendChar(c);
    }

    return ss;
}
```

`OutputDataReceived` `ErrorDataReceived` (`cmdOut` STDOUT STDERR .

: https://riptutorial.com/ko/dot-net/topic/3143/-

43:

.Net new () . . .Net Garbage Collector .

" " .

. .

- Collect () ( ) "optimized"
- Collect () AddMemoryPressure () RemoveMemoryPressure () .
- . ""3 .
- () .

Examples

()

:

```
public class FinalizableObject
{
    public FinalizableObject()
    {
        Console.WriteLine("Instance initialized");
    }

    ~FinalizableObject()
    {
        Console.WriteLine("Instance finalized");
    }
}
```

:

```
new FinalizableObject(); // Object instantiated, ready to be used
```

.

```
<namespace>.FinalizableObject initialized
```

. .

.

```
new FinalizableObject(); // Object instantiated, ready to be used
GC.Collect();
```

.

```
<namespace>.FinalizableObject initialized
<namespace>.FinalizableObject finalized
```

("") .

-

: " " ," " ( ) .

(FinalizableObject1 FinalizableObject2 FinalizableObject / .)

```
var obj1 = new FinalizableObject1(); // Finalizable1 instance allocated here
var obj2 = new FinalizableObject2(); // Finalizable2 instance allocated here
obj1 = null; // No more references to the Finalizable1 instance
GC.Collect();
```

.

```
<namespace>.FinalizableObject1 initialized
<namespace>.FinalizableObject2 initialized
<namespace>.FinalizableObject1 finalized
```

FinalizableObject1 Finalize FinalizableObject2 .

( ) ? OtherObject FinalizableObject public .

```
var obj1 = new FinalizableObject1();
var obj2 = new FinalizableObject2();
obj1.OtherObject = obj2;
obj2.OtherObject = obj1;
obj1 = null; // Program no longer references Finalizable1 instance
obj2 = null; // Program no longer references Finalizable2 instance
// But the two objects still reference each other
GC.Collect();
```

.

```
<namespace>.FinalizedObject1 initialized
<namespace>.FinalizedObject2 initialized
<namespace>.FinalizedObject1 finalized
<namespace>.FinalizedObject2 finalized
```

( ) .

("") "". , "" " " .

:

```
var weak = new WeakReference<FinalizableObject>(new FinalizableObject());
GC.Collect();
```

.

```
<namespace>.FinalizableObject initialized
<namespace>.FinalizableObject finalized
```

WeakReference ( ) .

1 : WeakReference .

2 : Weakreference . TryGetTarget.

```
var target = new object(); // Any object will do as target
var weak = new WeakReference<object>(target); // Create weak reference
target = null; // Drop strong reference to the target

// ... Many things may happen in-between

// Check whether the target is still available
if(weak.TryGetTarget(out target))
{
    // Use re-initialized target variable
    // To do whatever the target is needed for
}
else
{
    // Do something when there is no more target object
    // The target variable value should not be used here
}
```

WeakReference .Net 4.5 . , .

```
var target = new object(); // Any object will do as target
var weak = new WeakReference(target); // Create weak reference
target = null; // Drop strong reference to the target

// ... Many things may happen in-between

// Check whether the target is still available
if (weak.IsAlive)
{
    target = weak.Target;

    // Use re-initialized target variable
    // To do whatever the target is needed for
}
else
{
    // Do something when there is no more target object
    // The target variable value should not be used here
}
```

()

Dispose () IDisposable . "catch" Dispose () (finalizer).

Dispose () .

```
private void SomeFunction()
{
    // Initialize an object that uses heavy external resources
    var disposableObject = new ClassThatImplementsIDisposable();

    // ... Use that object

    // Dispose as soon as no longer used
    disposableObject.Dispose();

    // ... Do other stuff

    // The disposableObject variable gets out of scope here
    // The object will be finalized later on (no guarantee when)
    // But it no longer holds to the heavy external resource after it was disposed
}
```

. . MVC System.Web.Mvc.ControllerBase . IDisposable Dispose () . .

Dispose () . .

- finalizer .
- Dispose () .

Dispose () finalizer . .

-
- Dispose () finalizer

. .

- .
- " " .

. " " .

```
public class DisposableFinalizable1: IDisposable
{
    private bool disposed = false;

    ~DisposableFinalizable1() { Cleanup(); }

    public void Dispose() { Cleanup(); }

    private void Cleanup()
    {
        if(!disposed)
        {
            // Actual code to release resources gets here, then
            disposed = true;
        }
    }
}
```

Garbage Collector Dispose Finalizer SuppressFinalize () .

```
public class DisposableFinalizable2 : IDisposable
{
    ~DisposableFinalizable2() { Cleanup(); }

    public void Dispose()
    {
        Cleanup();
        GC.SuppressFinalize(this);
    }

    private void Cleanup()
    {
        // Actual code to release resources gets here
    }
}
```

: https://riptutorial.com/ko/dot-net/topic/9636/-

44: /

.NET Framework . , .

Examples

Rijndael Managed

: `System.Security.Cryptography`

```
private class Encryption {

          private const string SecretKey = "topSecretKeyusedforEncryptions";

          private const string SecretIv = "secretVectorHere";

          public string Encrypt(string data) {
            return string.IsNullOrEmpty(data) ? data :
Convert.ToBase64String(this.EncryptStringToBytesAes(data, this.GetCryptographyKey(),
this.GetCryptographyIv()));
          }

          public string Decrypt(string data) {
            return string.IsNullOrEmpty(data) ? data :
this.DecryptStringFromBytesAes(Convert.FromBase64String(data), this.GetCryptographyKey(),
this.GetCryptographyIv());
          }

          private byte[] GetCryptographyKey() {
            return Encoding.ASCII.GetBytes(SecretKey.Replace('e', '!'));
          }

          private byte[] GetCryptographyIv() {
            return Encoding.ASCII.GetBytes(SecretIv.Replace('r', '!'));
          }

          private byte[] EncryptStringToBytesAes(string plainText, byte[] key, byte[] iv) {
            MemoryStream encrypt;
            RijndaelManaged aesAlg = null;
            try {
              aesAlg = new RijndaelManaged {
                Key = key,
                IV = iv
              };
              var encryptor = aesAlg.CreateEncryptor(aesAlg.Key, aesAlg.IV);
              encrypt = new MemoryStream();
              using (var csEncrypt = new CryptoStream(encrypt, encryptor,
CryptoStreamMode.Write)) {
                using (var swEncrypt = new StreamWriter(csEncrypt)) {
                  swEncrypt.Write(plainText);
                }
              }
            } finally {
              aesAlg?.Clear();
            }
            return encrypt.ToArray();
```

```
          }

          private string DecryptStringFromBytesAes(byte[] cipherText, byte[] key, byte[] iv) {
            RijndaelManaged aesAlg = null;
            string plaintext;
            try {
              aesAlg = new RijndaelManaged {
                Key = key,
                IV = iv
              };
              var decryptor = aesAlg.CreateDecryptor(aesAlg.Key, aesAlg.IV);
              using (var msDecrypt = new MemoryStream(cipherText)) {
                using (var csDecrypt = new CryptoStream(msDecrypt, decryptor,
CryptoStreamMode.Read)) {
                  using (var srDecrypt = new StreamReader(csDecrypt))
                    plaintext = srDecrypt.ReadToEnd();
                }
              }
            } finally {
              aesAlg?.Clear();
            }
            return plaintext;
          }
        }
```

```
var textToEncrypt = "hello World";

 var encrypted = new Encryption().Encrypt(textToEncrypt); //-> zBmW+FUxOvdbpOGm9Ss/vQ==

 var decrypted = new Encryption().Decrypt(encrypted); //-> hello World
```

:

- Rijndael AES .

AES (C #)

```
using System;
using System.IO;
using System.Security.Cryptography;

namespace Aes_Example
{
    class AesExample
    {
        public static void Main()
        {
            try
            {
                string original = "Here is some data to encrypt!";

                // Create a new instance of the Aes class.
                // This generates a new key and initialization vector (IV).
                using (Aes myAes = Aes.Create())
                {
                    // Encrypt the string to an array of bytes.
                    byte[] encrypted = EncryptStringToBytes_Aes(original,
                                                                myAes.Key,
```

```
                                                        myAes.IV);

            // Decrypt the bytes to a string.
            string roundtrip = DecryptStringFromBytes_Aes(encrypted,
                                                          myAes.Key,
                                                          myAes.IV);

            //Display the original data and the decrypted data.
            Console.WriteLine("Original:   {0}", original);
            Console.WriteLine("Round Trip: {0}", roundtrip);
        }
    }
    catch (Exception e)
    {
        Console.WriteLine("Error: {0}", e.Message);
    }
}

static byte[] EncryptStringToBytes_Aes(string plainText, byte[] Key, byte[] IV)
{
    // Check arguments.
    if (plainText == null || plainText.Length <= 0)
        throw new ArgumentNullException("plainText");
    if (Key == null || Key.Length <= 0)
        throw new ArgumentNullException("Key");
    if (IV == null || IV.Length <= 0)
        throw new ArgumentNullException("IV");

    byte[] encrypted;

    // Create an Aes object with the specified key and IV.
    using (Aes aesAlg = Aes.Create())
    {
        aesAlg.Key = Key;
        aesAlg.IV = IV;

        // Create a decrytor to perform the stream transform.
        ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key,
                                                            aesAlg.IV);

        // Create the streams used for encryption.
        using (MemoryStream msEncrypt = new MemoryStream())
        {
            using (CryptoStream csEncrypt = new CryptoStream(msEncrypt,
                                                             encryptor,
                                                             CryptoStreamMode.Write))
            {
                using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))
                {
                    //Write all data to the stream.
                    swEncrypt.Write(plainText);
                }

                encrypted = msEncrypt.ToArray();
            }
        }
    }

    // Return the encrypted bytes from the memory stream.
    return encrypted;
}
```

```
        static string DecryptStringFromBytes_Aes(byte[] cipherText, byte[] Key, byte[] IV)
        {
            // Check arguments.
            if (cipherText == null || cipherText.Length <= 0)
                throw new ArgumentNullException("cipherText");
            if (Key == null || Key.Length <= 0)
                throw new ArgumentNullException("Key");
            if (IV == null || IV.Length <= 0)
                throw new ArgumentNullException("IV");

            // Declare the string used to hold the decrypted text.
            string plaintext = null;

            // Create an Aes object with the specified key and IV.
            using (Aes aesAlg = Aes.Create())
            {
                aesAlg.Key = Key;
                aesAlg.IV = IV;

                // Create a decrytor to perform the stream transform.
                ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key,
                                                                    aesAlg.IV);

                // Create the streams used for decryption.
                using (MemoryStream msDecrypt = new MemoryStream(cipherText))
                {
                    using (CryptoStream csDecrypt = new CryptoStream(msDecrypt,
                                                                     decryptor,
                                                                     CryptoStreamMode.Read))
                    {
                        using (StreamReader srDecrypt = new StreamReader(csDecrypt))
                        {

                            // Read the decrypted bytes from the decrypting stream
                            // and place them in a string.
                            plaintext = srDecrypt.ReadToEnd();
                        }
                    }
                }
            }

            return plaintext;
        }
    }
}
```

MSDN .

AES .

(**AES = Advanced Encryption Standard** , (NIST) 2001 .)

:

- (`CipherMode` `Mode`). `CipherMode.ECB` ( ) .
- ( ) `Key` ( ) . **KeySize** 256 . `LegalKeySizes` .
-

`IV` SALT (**Random SALT**)

- `SupportedBlockSizes` `BlockSize` .

: Main () .

/ SALT (C #)

```
using System;
using System.Security.Cryptography;
using System.Text;

public class PasswordDerivedBytesExample
{
    public static void Main(String[] args)
    {
        // Get a password from the user.
        Console.WriteLine("Enter a password to produce a key:");

        byte[] pwd = Encoding.Unicode.GetBytes(Console.ReadLine());

        byte[] salt = CreateRandomSalt(7);

        // Create a TripleDESCryptoServiceProvider object.
        TripleDESCryptoServiceProvider tdes = new TripleDESCryptoServiceProvider();

        try
        {
            Console.WriteLine("Creating a key with PasswordDeriveBytes...");

            // Create a PasswordDeriveBytes object and then create
            // a TripleDES key from the password and salt.
            PasswordDeriveBytes pdb = new PasswordDeriveBytes(pwd, salt);

            // Create the key and set it to the Key property
            // of the TripleDESCryptoServiceProvider object.
            tdes.Key = pdb.CryptDeriveKey("TripleDES", "SHA1", 192, tdes.IV);

            Console.WriteLine("Operation complete.");
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
        finally
        {
            // Clear the buffers
            ClearBytes(pwd);
            ClearBytes(salt);

            // Clear the key.
            tdes.Clear();
        }

        Console.ReadLine();
    }

    #region Helper methods
```

```
    /// <summary>
    /// Generates a random salt value of the specified length.
    /// </summary>
    public static byte[] CreateRandomSalt(int length)
    {
        // Create a buffer
        byte[] randBytes;

        if (length >= 1)
        {
            randBytes = new byte[length];
        }
        else
        {
            randBytes = new byte[1];
        }

        // Create a new RNGCryptoServiceProvider.
        RNGCryptoServiceProvider rand = new RNGCryptoServiceProvider();

        // Fill the buffer with random bytes.
        rand.GetBytes(randBytes);

        // return the bytes.
        return randBytes;
    }

    /// <summary>
    /// Clear the bytes in a buffer so they can't later be read from memory.
    /// </summary>
    public static void ClearBytes(byte[] buffer)
    {
        // Check arguments.
        if (buffer == null)
        {
            throw new ArgumentNullException("buffer");
        }

        // Set each byte in the buffer to 0.
        for (int x = 0; x < buffer.Length; x++)
        {
            buffer[x] = 0;
        }
    }

    #endregion
}
```

MSDN .

SALT .

:

- `PasswordDeriveBytes` PBKDF1 . 100 . SALT .
- `CryptDeriveKey` ("SHA1") `PasswordDeriveBytes` ("TripleDES") . keysize 192 , IV DES
-

. ( ) .

- `CryptDeriveKey` AES . .
  : AES AES `CryptDeriveKey` TripleDES-Container AES . TripleDES AES TripleDES .

: Main () .

(AES)

```
    public static string Decrypt(string cipherText)
    {
        if (cipherText == null)
            return null;

        byte[] cipherBytes = Convert.FromBase64String(cipherText);
        using (Aes encryptor = Aes.Create())
        {
            Rfc2898DeriveBytes pdb = new Rfc2898DeriveBytes(CryptKey, new byte[] { 0x49, 0x76,
0x61, 0x6e, 0x20, 0x4d, 0x65, 0x64, 0x76, 0x65, 0x64, 0x65, 0x76 });
            encryptor.Key = pdb.GetBytes(32);
            encryptor.IV = pdb.GetBytes(16);

            using (MemoryStream ms = new MemoryStream())
            {
                using (CryptoStream cs = new CryptoStream(ms, encryptor.CreateDecryptor(),
CryptoStreamMode.Write))
                {
                    cs.Write(cipherBytes, 0, cipherBytes.Length);
                    cs.Close();
                }

                cipherText = Encoding.Unicode.GetString(ms.ToArray());
            }
        }

        return cipherText;
    }
```

```
    public static string Encrypt(string cipherText)
    {
        if (cipherText == null)
            return null;

        byte[] clearBytes = Encoding.Unicode.GetBytes(cipherText);
        using (Aes encryptor = Aes.Create())
        {
            Rfc2898DeriveBytes pdb = new Rfc2898DeriveBytes(CryptKey, new byte[] { 0x49, 0x76,
0x61, 0x6e, 0x20, 0x4d, 0x65, 0x64, 0x76, 0x65, 0x64, 0x65, 0x76 });
            encryptor.Key = pdb.GetBytes(32);
            encryptor.IV = pdb.GetBytes(16);

            using (MemoryStream ms = new MemoryStream())
            {
                using (CryptoStream cs = new CryptoStream(ms, encryptor.CreateEncryptor(),
CryptoStreamMode.Write))
                {
                    cs.Write(clearBytes, 0, clearBytes.Length);
```

```
                    cs.Close();
                }

                cipherText = Convert.ToBase64String(ms.ToArray());
            }
        }
        return cipherText;
    }
```

```
var textToEncrypt = "TestEncrypt";

var encrypted = Encrypt(textToEncrypt);

var decrypted = Decrypt(encrypted);
```

/ : https://riptutorial.com/ko/dot-net/topic/7615/---

45:

:

- MSDN : (C #)
- MSDN : Throw
- MSDN : CA1031 : catch
- MSDN : try-catch (C #)

Examples

throw . .

- .
- .
- 0 .
-

"catch" .

- .
- (catch)

.

`try { ... }  catch (ExceptionType) { ... }  .`

```
Console.Write("Please enter a filename: ");
string filename = Console.ReadLine();

Stream fileStream;

try
{
    fileStream = File.Open(filename);
}
catch (FileNotFoundException)
{
    Console.WriteLine("File '{0}' could not be found.", filename);
}
```

finally

`try-finally try-catch-finally finally { ... }` (: `StackOverflowException` throw `Environment.FailFast()` `Environment.FailFast()`).

`try { ... }` .

```
Console.Write("Please enter a filename: ");
string filename = Console.ReadLine();
```

```
Stream fileStream = null;

try
{
    fileStream = File.Open(filename);
}
catch (FileNotFoundException)
{
    Console.WriteLine("File '{0}' could not be found.", filename);
}
finally
{
    if (fileStream != null)
    {
        fileStream.Dispose();
    }
}
```

catch

throw . .

```
private static void AskTheUltimateQuestion()
{
    try
    {
        var x = 42;
        var y = x / (x - x); // will throw a DivideByZeroException

        // IMPORTANT NOTE: the error in following string format IS intentional
        // and exists to throw an exception to the FormatException catch, below
        Console.WriteLine("The secret to life, the universe, and everything is {1}", y);
    }
    catch (DivideByZeroException)
    {
        // we do not need a reference to the exception
        Console.WriteLine("Dividing by zero would destroy the universe.");

        // do this to preserve the stack trace:
        throw;
    }
    catch (FormatException ex)
    {
        // only do this if you need to change the type of the Exception to be thrown
        // and wrap the inner Exception

        // remember that the stack trace of the outer Exception will point to the
        // next line

        // you'll need to examine the InnerException property to get the stack trace
        // to the line that actually started the problem

        throw new InvalidOperationException("Watch your format string indexes.", ex);
    }
    catch (Exception ex)
    {
        Console.WriteLine("Something else horrible happened. The exception: " + ex.Message);
```

```
        // do not do this, because the stack trace will be changed to point to
        // this location instead of the location where the exception
        // was originally thrown:
        throw ex;
    }
}

static void Main()
{
    try
    {
        AskTheUltimateQuestion();
    }
    catch
    {
        // choose this kind of catch if you don't need any information about
        // the exception that was caught

        // this block "eats" all exceptions instead of rethrowing them
    }
}
```

(C # 6.0 , VB.NET ( )) .

/ C # /

C # 6.0 `when` .

`if` `when` .

```
try
{
  // ...
}
catch (Exception e) when (e.InnerException != null) // Any condition can go in here.
{
  // ...
}
```

C # 6.0 :

catch

`catch` `throw` *throw* . .

```
try {...}
catch (Exception ex) {
  // Note: the ex variable is *not* used
  throw;
}
```

`throw ex` `throw ex` .

```
try {...}
catch (Exception ex) {
```

```
  // Note: the ex variable is thrown
  //  future stack traces of the exception will not see prior calls
  throw ex;
}
```

throw ex throw ex . ex throw " " .

.

. ExceptionDispatchInfo ExceptionDispatchInfo .

```
using System.Runtime.ExceptionServices;

void Main()
{
    ExceptionDispatchInfo capturedException = null;
    try
    {
        throw new Exception();
    }
    catch (Exception ex)
    {
        capturedException = ExceptionDispatchInfo.Capture(ex);
    }

    Foo(capturedException);
}

void Foo(ExceptionDispatchInfo exceptionDispatchInfo)
{
    // Do stuff

    if (capturedException != null)
    {
        // Exception stack trace will show it was thrown from Main() and not from Foo()
        exceptionDispatchInfo.Throw();
    }
}
```

: https://riptutorial.com/ko/dot-net/topic/33/

46:

Examples

.Net

JIT GC .

CLR :
:
EE :
JIT : Just-In-Time
GC :
OOM :
STA :
MTA :

: https://riptutorial.com/ko/dot-net/topic/10939/

47:

`Greeter` .

```
public class ControlFreakGreeter
{
    public void Greet()
    {
        var greetingProvider = new SqlGreetingProvider(
            ConfigurationManager.ConnectionStrings["myConnectionString"].ConnectionString);
        var greeting = greetingProvider.GetGreeting();
        Console.WriteLine(greeting);
    }
}
```

, SQL , " ".

`Greeter` .

Dependency Inversion Principle ( ) . () . .

. `Greeter` `IGreetingProvider` `IGreetingWriter` " `IGreetingWriter` ." . , `IGreetingProvider` `IGreetingWriter` `IGreetingWriter` `Greeter` . . `Greeter` .

`ControlFreakGreeter` . SQL . . ControlFreakGreeter .

`Greeter` . app.config .

`IGreetingProvider` `IGreetingWriter` `IGreetingWriter` . . SQL `SqlGreetingProvider` . "". " " .

Examples

-

`Greeter` . . . . . , `IGreetingProvider` `IGreetingWriter` . `Greeter` "". ( )

```
public class Greeter
{
    private readonly IGreetingProvider _greetingProvider;
    private readonly IGreetingWriter _greetingWriter;

    public Greeter(IGreetingProvider greetingProvider, IGreetingWriter greetingWriter)
    {
        _greetingProvider = greetingProvider;
        _greetingWriter = greetingWriter;
    }

    public void Greet()
    {
        var greeting = _greetingProvider.GetGreeting();
        _greetingWriter.WriteGreeting(greeting);
```

```
    }
}

public interface IGreetingProvider
{
    string GetGreeting();
}

public interface IGreetingWriter
{
    void WriteGreeting(string greeting);
}
```

Greeting IGreetingProvider IGreetingWriter , . .Greeting . "" .

"".

:

- private . .
- private readonly . . . .
- . . .

IGreetingProvider IGreetingWriter Greeter .

IGreetingProvider API .IGreetingWriter . Greeter . Moq , .

```
public class TestGreetingProvider : IGreetingProvider
{
    public const string TestGreeting = "Hello!";

    public string GetGreeting()
    {
        return TestGreeting;
    }
}

public class TestGreetingWriter : List<string>, IGreetingWriter
{
    public void WriteGreeting(string greeting)
    {
        Add(greeting);
    }
}

[TestClass]
public class GreeterTests
{
    [TestMethod]
    public void Greeter_WritesGreeting()
    {
        var greetingProvider = new TestGreetingProvider();
        var greetingWriter = new TestGreetingWriter();
        var greeter = new Greeter(greetingProvider, greetingWriter);
        greeter.Greet();
```

```
        Assert.AreEqual(greetingWriter[0], TestGreetingProvider.TestGreeting);
    }
}
```

`IGreetingProvider` `IGreetingWriter` . Google `Greeter` `Greeter.Greeter` ( ) . `Greeter` .

(IoC)

("").

Castle Windsor, Autofac, SimpleInjector, Ninject, Unity ("DI ", "IoC " "") .

. , . , . . , `IDisposable` .

. : `.CustomerService` . . :

```
public CustomerData GetCustomerData(string customerNumber)
{
    var customerApiEndpoint =
ConfigurationManager.AppSettings["customerApi:customerApiEndpoint"];
    var logFilePath = ConfigurationManager.AppSettings["logwriter:logFilePath"];
    var authConnectionString =
ConfigurationManager.ConnectionStrings["authorization"].ConnectionString;
    using(var logWriter = new LogWriter(logFilePath ))
    {
        using(var customerApiClient = new CustomerApiClient(customerApiEndpoint))
        {
            var customerService = new CustomerService(
                new SqlAuthorizationRepository(authorizationConnectionString, logWriter),
                new CustomerDataRepository(customerApiClient, logWriter),
                logWriter
            );

            // All this just to create an instance of CustomerService!
            return customerService.GetCustomerData(string customerNumber);
        }
    }
}
```

`CustomerService` ? `IDisposable` . . `CustomerService` `GetCustomerService()` .

`IDisposable` ? . . , . . .

. . Castle Windsor .

```
var container = new WindsorContainer()
container.Register(
    Component.For<CustomerService>(),
    Component.For<ILogWriter, LogWriter>()
        .DependsOn(Dependency.OnAppSettingsValue("logFilePath", "logWriter:logFilePath")),
    Component.For<IAuthorizationRepository, SqlAuthorizationRepository>()
        .DependsOn(Dependency.OnValue(connectionString,
ConfigurationManager.ConnectionStrings["authorization"].ConnectionString)),
    Component.For<ICustomerDataProvider, CustomerApiClient>()
```

```
        .DependsOn(Dependency.OnAppSettingsValue("apiEndpoint",
"customerApi:customerApiEndpoint"))
);
```

" " " ". WindsorContainer .

- ILogWriter LogWriter . LogWriter . AppSettings .
- IAuthorizationRepository SqlAuthorizationRepository SqlAuthorizationRepository . . ConnectionStrings .
- ICustomerDataProvider CustomerApiClient AppSettings .

"". . .

```
var customerService = container.Resolve<CustomerService>();
var data = customerService.GetCustomerData(customerNumber);
container.Release(customerService);
```

CustomerService IAuthorizationRepository ICustomerDataProvider IAuthorizationRepository .
. . CustomerService .

IDoesSomethingElse CustomerService .

DI . , LogWriter ILogWriter , ? .

IDisposable ? container.Release(customerService); container.Release(customerService); .
(Windsor) Dispose Dispose . CustomerService IDisposable .

. . .

. . .

: https://riptutorial.com/ko/dot-net/topic/5085/-

48: (TPL)

.

* :

- UI
-
- (, TCP / UDP)
-

* . .

TPL with .Net 3.5

TPL NuGet .Net 3.5 .

Examples

- (BlockingCollection)

```
var collection = new BlockingCollection<int>(5);
var random = new Random();

var producerTask = Task.Run(() => {
    for(int item=1; item<=10; item++)
    {
        collection.Add(item);
        Console.WriteLine("Produced: " + item);
        Thread.Sleep(random.Next(10,1000));
    }
    collection.CompleteAdding();
    Console.WriteLine("Producer completed!");
});
```

`collection.CompleteAdding();` `collection.CompleteAdding();` . `collection.CompleteAdding();` `collection.CompleteAdding();` . BlockingCollection . BlockingCollection `collection.GetConsumingEnumerable()` Enumerable BlockingCollection.CompleteAdding (); .

```
var consumerTask = Task.Run(() => {
    foreach(var item in collection.GetConsumingEnumerable())
    {
        Console.WriteLine("Consumed: " + item);
        Thread.Sleep(random.Next(10,1000));
    }
    Console.WriteLine("Consumer completed!");
});

Task.WaitAll(producerTask, consumerTask);
```

```
Console.WriteLine("Everything completed!");
```

:

Task Task ...

```
var task = new Task(() =>
{
    Console.WriteLine("Task code starting...");
    Thread.Sleep(2000);
    Console.WriteLine("...task code ending!");
});

Console.WriteLine("Starting task...");
task.Start();
task.Wait();
Console.WriteLine("Task completed!");
```

... Task.Run :

```
Console.WriteLine("Starting task...");
var task = Task.Run(() =>
{
    Console.WriteLine("Task code starting...");
    Thread.Sleep(2000);
    Console.WriteLine("...task code ending!");
});
task.Wait();
Console.WriteLine("Task completed!");
```

Start .

: WaitAll

```
var tasks = Enumerable.Range(1, 5).Select(n => new Task<int>(() =>
{
    Console.WriteLine("I'm task " + n);
    return n;
})).ToArray();

foreach(var task in tasks) task.Start();
Task.WaitAll(tasks);

foreach(var task in tasks)
    Console.WriteLine(task.Result);
```

: WaitAny

```
var allTasks = Enumerable.Range(1, 5).Select(n => new Task<int>(() => n)).ToArray();
var pendingTasks = allTasks.ToArray();

foreach(var task in allTasks) task.Start();

while(pendingTasks.Length > 0)
```

```
{
    var finishedTask = pendingTasks[Task.WaitAny(pendingTasks)];
    Console.WriteLine("Task {0} finished", finishedTask.Result);
    pendingTasks = pendingTasks.Except(new[] {finishedTask}).ToArray();
}

Task.WaitAll(allTasks);
```

: `WaitAny` `WaitAll` .

: (Wait)

```
var task1 = Task.Run(() =>
{
    Console.WriteLine("Task 1 code starting...");
    throw new Exception("Oh no, exception from task 1!!");
});

var task2 = Task.Run(() =>
{
    Console.WriteLine("Task 2 code starting...");
    throw new Exception("Oh no, exception from task 2!!");
});

Console.WriteLine("Starting tasks...");
try
{
    Task.WaitAll(task1, task2);
}
catch(AggregateException ex)
{
    Console.WriteLine("Task(s) failed!");
    foreach(var inner in ex.InnerExceptions)
        Console.WriteLine(inner.Message);
}

Console.WriteLine("Task 1 status is: " + task1.Status); //Faulted
Console.WriteLine("Task 2 status is: " + task2.Status); //Faulted
```

: ( )

```
var task1 = Task.Run(() =>
{
    Console.WriteLine("Task 1 code starting...");
    throw new Exception("Oh no, exception from task 1!!");
});

var task2 = Task.Run(() =>
{
    Console.WriteLine("Task 2 code starting...");
    throw new Exception("Oh no, exception from task 2!!");
});

var tasks = new[] {task1, task2};

Console.WriteLine("Starting tasks...");
while(tasks.All(task => !task.IsCompleted));
```

```
foreach(var task in tasks)
{
    if(task.IsFaulted)
        Console.WriteLine("Task failed: " +
            task.Exception.InnerExceptions.First().Message);
}

Console.WriteLine("Task 1 status is: " + task1.Status); //Faulted
Console.WriteLine("Task 2 status is: " + task2.Status); //Faulted
```

: CancellationToken

```
var cancellationTokenSource = new CancellationTokenSource();
var cancellationToken = cancellationTokenSource.Token;

var task = new Task((state) =>
    {
        int i = 1;
        var myCancellationToken = (CancellationToken)state;
        while(true)
        {
            Console.Write("{0} ", i++);
            Thread.Sleep(1000);
            myCancellationToken.ThrowIfCancellationRequested();
        }
    },
    cancellationToken: cancellationToken,
    state: cancellationToken);

Console.WriteLine("Counting to infinity. Press any key to cancel!");
task.Start();
Console.ReadKey();

cancellationTokenSource.Cancel();
try
{
    task.Wait();
}
catch(AggregateException ex)
{
    ex.Handle(inner => inner is OperationCanceledException);
}

Console.WriteLine($"{Environment.NewLine}You have cancelled! Task status is: {task.Status}");
//Canceled
```

`ThrowIfCancellationRequested` , `IsCancellationRequested` `OperationCanceledException` throw .

```
//New task delegate
int i = 1;
var myCancellationToken = (CancellationToken)state;
while(!myCancellationToken.IsCancellationRequested)
{
    Console.Write("{0} ", i++);
    Thread.Sleep(1000);
}
Console.WriteLine($"{Environment.NewLine}Ouch, I have been cancelled!!");
```

```
throw new OperationCanceledException(myCancellationToken);
```

cancellationToken . ThrowIfCancellationRequested ThrowIfCancellationRequested Faulted Canceled . OperationCanceledException .

Task.WhenAny

```
var random = new Random();
IEnumerable<Task<int>> tasks = Enumerable.Range(1, 5).Select(n => Task.Run(async() =>
{
    Console.WriteLine("I'm task " + n);
    await Task.Delay(random.Next(10,1000));
    return n;
}));

Task<Task<int>> whenAnyTask = Task.WhenAny(tasks);
Task<int> completedTask = await whenAnyTask;
Console.WriteLine("The winner is: task " + await completedTask);

await Task.WhenAll(tasks);
Console.WriteLine("All tasks finished!");
```

Task.WhenAll

```
var random = new Random();
IEnumerable<Task<int>> tasks = Enumerable.Range(1, 5).Select(n => Task.Run(() =>
{
    Console.WriteLine("I'm task " + n);
    return n;
}));

Task<int[]> task = Task.WhenAll(tasks);
int[] results = await task;

Console.WriteLine(string.Join(",", results.Select(n => n.ToString())));
// Output: 1,2,3,4,5
```

Parallel.Invoke

```
var actions = Enumerable.Range(1, 10).Select(n => new Action(() =>
{
    Console.WriteLine("I'm task " + n);
    if((n & 1) == 0)
        throw new Exception("Exception from task " + n);
})).ToArray();

try
{
    Parallel.Invoke(actions);
}
catch(AggregateException ex)
{
    foreach(var inner in ex.InnerExceptions)
        Console.WriteLine("Task failed: " + inner.Message);
}
```

Parallel.ForEach

`Parallel.ForEach` 1 10000 . `Interlocked.Add` .

```
using System.Threading;

int Foo()
{
    int total = 0;
    var numbers = Enumerable.Range(1, 10000).ToList();
    Parallel.ForEach(numbers,
        () => 0, // initial value,
        (num, state, localSum) => num + localSum,
        localSum => Interlocked.Add(ref total, localSum));
    return total; // total = 50005000
}
```

Parallel.For

`Parallel.For` 1 10000 . `Interlocked.Add` .

```
using System.Threading;

int Foo()
{
    int total = 0;
    Parallel.For(1, 10001,
        () => 0, // initial value,
        (num, state, localSum) => num + localSum,
        localSum => Interlocked.Add(ref total, localSum));
    return total; // total = 50005000
}
```

AsyncLocal

`AsyncLocal` `AsyncLocal` .

```
void Main()
{
    AsyncLocal<string> user = new AsyncLocal<string>();
    user.Value = "initial user";

    // this does not affect other tasks - values are local relative to the branches of
execution flow
    Task.Run(() => user.Value = "user from another task");

    var task1 = Task.Run(() =>
    {
        Console.WriteLine(user.Value); // outputs "initial user"
        Task.Run(() =>
        {
            // outputs "initial user" - value has flown from main method to this task without
being changed
            Console.WriteLine(user.Value);
        }).Wait();
```

```
        user.Value = "user from task1";

        Task.Run(() =>
        {
            // outputs "user from task1" - value has flown from main method to task1
            // than value was changed and flown to this task.
            Console.WriteLine(user.Value);
        }).Wait();
    });

    task1.Wait();

    // ouputs "initial user" - changes do not propagate back upstream the execution flow
    Console.WriteLine(user.Value);
}
```

: `AsynLocal.Value copy on read` semantic . `AsyncLocal` .

VB.NET Parallel.ForEach

```
For Each row As DataRow In FooDataTable.Rows
    Me.RowsToProcess.Add(row)
Next

Dim myOptions As ParallelOptions = New ParallelOptions()
myOptions.MaxDegreeOfParallelism = environment.processorcount

Parallel.ForEach(RowsToProcess, myOptions, Sub(currentRow, state)
                                               ProcessRowParallel(currentRow, state)
                                           End Sub)
```

:

`Task< TResult >` TResult . Result .

```
Task<int> t = Task.Run(() =>
    {
        int sum = 0;

        for(int i = 0; i < 500; i++)
            sum += i;

        return sum;
    });

Console.WriteLine(t.Result); // Outuput 124750
```

.

```
public async Task DoSomeWork()
{
    WebClient client = new WebClient();
    // Because the task is awaited, result of the task is assigned to response
    string response = await client.DownloadStringTaskAsync("http://somedomain.com");
```

```
}
```

(TPL) : https://riptutorial.com/ko/dot-net/topic/55/----tpl-

49: (TPL) API

API . .. TPL .Net 4 .

TPL , , , .

Examples

UI

```
void MyButton_OnClick(object sender, EventArgs args)
{
    Task.Run(() => // Schedule work using the thread pool
        {
            System.Threading.Thread.Sleep(5000); // Sleep for 5 seconds to simulate work.
        })
    .ContinueWith(p => // this continuation contains the 'update' code to run on the UI thread
    {
        this.TextBlock_ResultText.Text = "The work completed at " + DateTime.Now.ToString()
    },
    TaskScheduler.FromCurrentSynchronizationContext()); // make sure the update is run on the
UI thread.

}
```

(TPL) API : https://riptutorial.com/ko/dot-net/topic/5164/----tpl--api-

50: (System.Text.RegularExpressions)

Examples

.

```
public bool Check()
{
    string input = "Hello World!";
    string pattern = @"H.ll. W.rld!";

    // true
    return Regex.IsMatch(input, pattern);
}
```

```
public bool Check()
{
    string input = "Hello World!";
    string pattern = @"H.ll. W.rld!";

    // true
    return Regex.IsMatch(input, pattern, RegexOptions.IgnoreCase | RegexOptions.Singleline);
}
```

```
public string Check()
{
    string input = "Hello World!";
    string pattern = @"W.rld";

    // Hello Stack Overflow!
    return Regex.Replace(input, pattern, "Stack Overflow");
}
```

```
public string Check()
{
    string input = "Hello World!";
    string pattern = @"H.ll. (?<Subject>W.rld)!";

    Match match = Regex.Match(input, pattern);

    // World
    return match.Groups["Subject"].Value;
}
```

.

```
public string Remove()
{
    string input = "Hello./!";

    return Regex.Replace(input, "[^a-zA-Z0-9]", "");
```

```
}
```

```
using System.Text.RegularExpressions;
```

```
static void Main(string[] args)
{
    string input = "Carrot Banana Apple Cherry Clementine Grape";
    // Find words that start with uppercase 'C'
    string pattern = @"\bC\w*\b";

    MatchCollection matches = Regex.Matches(input, pattern);
    foreach (Match m in matches)
        Console.WriteLine(m.Value);
}
```

```
Carrot
Cherry
Clementine
```

(System.Text.RegularExpressions) : https://riptutorial.com/ko/dot-net/topic/6944/--system-text-regularexpressions-

51:

Examples

```
var serialPort = new SerialPort("COM1", 9600, Parity.Even, 8, StopBits.One);
serialPort.Open();
serialPort.WriteLine("Test data");
string response = serialPort.ReadLine();
Console.WriteLine(response);
serialPort.Close();
```

```
string[] portNames = SerialPort.GetPortNames();
```

```
void SetupAsyncRead(SerialPort serialPort)
{
    serialPort.DataReceived += (sender, e) => {
        byte[] buffer = new byte[4096];
        switch (e.EventType)
        {
            case SerialData.Chars:
                var port = (SerialPort)sender;
                int bytesToRead = port.BytesToRead;
                if (bytesToRead > buffer.Length)
                    Array.Resize(ref buffer, bytesToRead);
                int bytesRead = port.Read(buffer, 0, bytesToRead);
                // Process the read buffer here
                // ...
                break;
            case SerialData.Eof:
                // Terminate the service here
                // ...
                break;
        }
    };
```

```
using System.IO.Ports;

namespace TextEchoService
{
    class Program
    {
        static void Main(string[] args)
        {
            var serialPort = new SerialPort("COM1", 9600, Parity.Even, 8, StopBits.One);
            serialPort.Open();
            string message = "";
            while (message != "quit")
            {
                message = serialPort.ReadLine();
                serialPort.WriteLine(message);
            }
            serialPort.Close();
        }
    }
```

```
}
```

```
using System;
using System.Collections.Generic;
using System.IO.Ports;
using System.Text;
using System.Threading;

namespace AsyncReceiver
{
    class Program
    {
        const byte STX = 0x02;
        const byte ETX = 0x03;
        const byte ACK = 0x06;
        const byte NAK = 0x15;
        static ManualResetEvent terminateService = new ManualResetEvent(false);
        static readonly object eventLock = new object();
        static List<byte> unprocessedBuffer = null;

        static void Main(string[] args)
        {
            try
            {
                var serialPort = new SerialPort("COM11", 9600, Parity.Even, 8, StopBits.One);
                serialPort.DataReceived += DataReceivedHandler;
                serialPort.ErrorReceived += ErrorReceivedHandler;
                serialPort.Open();
                terminateService.WaitOne();
                serialPort.Close();
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception occurred: {0}", e.Message);
            }
            Console.ReadKey();
        }

        static void DataReceivedHandler(object sender, SerialDataReceivedEventArgs e)
        {
            lock (eventLock)
            {
                byte[] buffer = new byte[4096];
                switch (e.EventType)
                {
                    case SerialData.Chars:
                        var port = (SerialPort)sender;
                        int bytesToRead = port.BytesToRead;
                        if (bytesToRead > buffer.Length)
                            Array.Resize(ref buffer, bytesToRead);
                        int bytesRead = port.Read(buffer, 0, bytesToRead);
                        ProcessBuffer(buffer, bytesRead);
                        break;
                    case SerialData.Eof:
                        terminateService.Set();
                        break;
                }
            }
        }
        static void ErrorReceivedHandler(object sender, SerialErrorReceivedEventArgs e)
```

```
        {
            lock (eventLock)
                if (e.EventType == SerialError.TXFull)
                {
                    Console.WriteLine("Error: TXFull. Can't handle this!");
                    terminateService.Set();
                }
                else
                {
                    Console.WriteLine("Error: {0}. Resetting everything", e.EventType);
                    var port = (SerialPort)sender;
                    port.DiscardInBuffer();
                    port.DiscardOutBuffer();
                    unprocessedBuffer = null;
                    port.Write(new byte[] { NAK }, 0, 1);
                }
        }

        static void ProcessBuffer(byte[] buffer, int length)
        {
            List<byte> message = unprocessedBuffer;
            for (int i = 0; i < length; i++)
                if (buffer[i] == ETX)
                {
                    if (message != null)
                    {
                        Console.WriteLine("MessageReceived: {0}",
                            Encoding.ASCII.GetString(message.ToArray()));
                        message = null;
                    }
                }
                else if (buffer[i] == STX)
                    message = null;
                else if (message != null)
                    message.Add(buffer[i]);
            unprocessedBuffer = message;
        }
    }
}
```

`STX` `ETX` . . . .

.

- (`SerialPort.DataReceived`).
- (`SerialPort.ErrorReceived`).
- .
- .
  - `SerialPort.DataReceived` (`ETX`) . (SerialPort.Read (..., ..., port.BytesToRead)). (`unprocessedBuffer`) .
- .
  - `SerialPort.DataReceived` .

: https://riptutorial.com/ko/dot-net/topic/5366/-

52: IProgress

Examples

IProgress<T> . .

```
void Main()
{
    IProgress<int> p = new Progress<int>(progress =>
    {
        Console.WriteLine("Running Step: {0}", progress);
    });
    LongJob(p);
}

public void LongJob(IProgress<int> progress)
{
    var max = 10;
    for (int i = 0; i < max; i++)
    {
        progress.Report(i);
    }
}
```

:

```
Running Step: 0
Running Step: 3
Running Step: 4
Running Step: 5
Running Step: 6
Running Step: 7
Running Step: 8
Running Step: 9
Running Step: 2
Running Step: 1
```

. IProgress<T>.Report() .

IProgress

System.Progress<T> Report() . IProgress<T> IProgress<T> Progress<T> .

```
var p1 = new Progress<int>();
p1.Report(1); //compiler error, Progress does not contain method 'Report'

IProgress<int> p2 = new Progress<int>();
p2.Report(2); //works

var p3 = new Progress<int>();
((IProgress<int>)p3).Report(3); //works
```

IProgress : https://riptutorial.com/ko/dot-net/topic/5628/---t---iprogress--t-

53:

.

- Array
- List
- Queue
- SortedList
- Stack
-

Examples

```
public class Model
{
    public string Name { get; set; }
    public bool? Selected { get; set; }
}
```

. Name nullable Selected . List<Model> .

```
var SelectedEmployees = new List<Model>
 {
      new Model() {Name = "Item1", Selected = true},
      new Model() {Name = "Item2", Selected = false},
      new Model() {Name = "Item3", Selected = false},
      new Model() {Name = "Item4"}
 };
```

Model new . ?

```
public class Model
{

    public Model(string name, bool? selected = false)
    {
        Name = name;
        selected = Selected;
    }
    public string Name { get; set; }
    public bool? Selected { get; set; }
}
```

.

```
var SelectedEmployees = new List<Model>
{
    new Model("Mark", true),
    new Model("Alexis"),
    new Model("")
};
```

?

```
public class Model
{
    public string Name { get; set; }
    public bool? Selected { get; set; }
}

public class ExtendedModel : Model
{
    public ExtendedModel()
    {
        BaseModel = new Model();
    }

    public Model BaseModel { get; set; }
    public DateTime BirthDate { get; set; }
}
```

`Model` .

```
var SelectedWithBirthDate = new List<ExtendedModel>
{
    new ExtendedModel()
    {
        BaseModel = new Model { Name = "Mark", Selected = true},
        BirthDate = new DateTime(2015, 11, 23)
    },
                    new ExtendedModel()
    {
        BaseModel = new Model { Name = "Random"},
        BirthDate = new DateTime(2015, 11, 23)
    }
};
```

`List<ExtendedModel>` `Collection<ExtendedModel>` , `ExtendedModel[]` , `object[]` `[]` .

FIFO (first-in first-out) `Queue` .Net . `Enqueue(T item)` `Dequeue()` . .

.

```
using System.Collections.Generic;
```

:

```
Queue<string> queue = new Queue<string>();
queue.Enqueue("John");
queue.Enqueue("Paul");
queue.Enqueue("George");
queue.Enqueue("Ringo");

string dequeueValue;
dequeueValue = queue.Dequeue(); // return John
dequeueValue = queue.Dequeue(); // return Paul
dequeueValue = queue.Dequeue(); // return George
dequeueValue = queue.Dequeue(); // return Ringo
```

.

.

```
using System.Collections;
```

:

```
Queue queue = new Queue();
queue.Enqueue("Hello World"); // string
queue.Enqueue(5); // int
queue.Enqueue(1d); // double
queue.Enqueue(true); // bool
queue.Enqueue(new Product()); // Product object

object dequeueValue;
dequeueValue = queue.Dequeue(); // return Hello World (string)
dequeueValue = queue.Dequeue(); // return 5 (int)
dequeueValue = queue.Dequeue(); // return 1d (double)
dequeueValue = queue.Dequeue(); // return true (bool)
dequeueValue = queue.Dequeue(); // return Product (Product type)
```

Peek () .

```
Queue<int> queue = new Queue<int>();
queue.Enqueue(10);
queue.Enqueue(20);
queue.Enqueue(30);
queue.Enqueue(40);
queue.Enqueue(50);

foreach (int element in queue)
{
    Console.WriteLine(i);
}
```

( ) :

```
10
20
30
40
50
```

LIFO ( ) `Stack` .NET . `Push(T item)` `Pop()` . .

.

```
using System.Collections.Generic;
```

:

```
Stack<string> stack = new Stack<string>();
```

```
stack.Push("John");
stack.Push("Paul");
stack.Push("George");
stack.Push("Ringo");

string value;
value = stack.Pop(); // return Ringo
value = stack.Pop(); // return George
value = stack.Pop(); // return Paul
value = stack.Pop(); // return John
```

.

.

```
using System.Collections;
```

:

```
Stack stack = new Stack();
stack.Push("Hello World"); // string
stack.Push(5); // int
stack.Push(1d); // double
stack.Push(true); // bool
stack.Push(new Product()); // Product object

object value;
value = stack.Pop(); // return Product (Product type)
value = stack.Pop(); // return true (bool)
value = stack.Pop(); // return 1d (double)
value = stack.Pop(); // return 5 (int)
value = stack.Pop(); // return Hello World (string)
```

`Stack` Peek () .

```
Stack<int> stack = new Stack<int>();
stack.Push(10);
stack.Push(20);

var lastValueAdded = stack.Peek(); // 20
```

(LIFO) .

```
Stack<int> stack = new Stack<int>();
stack.Push(10);
stack.Push(20);
stack.Push(30);
stack.Push(40);
stack.Push(50);

foreach (int element in stack)
{
   Console.WriteLine(element);
}
```

( ) :

```
50
40
30
20
10
```

. , numbers .

```
List<int> numbers = new List<int>(){10, 9, 8, 7, 7, 6, 5, 10, 4, 3, 2, 1};
```

C # Add . Add .

Stack<T> Queue<T> .

/ Dictionary<TKey, TValue> / .

```
Dictionary<int, string> employee = new Dictionary<int, string>()
     {{44, "John"}, {45, "Bob"}, {47, "James"}, {48, "Franklin"}};
```

.

: https://riptutorial.com/ko/dot-net/topic/30/

54:

/ . . .

Examples

.

...

```
void DoWork(string input)
{
    Contract.Requires(!string.IsNullOrEmpty(input));

    //do work
}
```

...

```
string s = null;
p.DoWork(s);
```

CodeContracts: requires is false: !string.IsNullOrEmpty(input)

. . .

...

```
string GetValue()
{
    Contract.Ensures(Contract.Result<string>() != null);

    return null;
}
```

...

```
string GetValue()
{
    Contract.Ensures(Contract.Result<string>() != null);

    return null;
}
```

CodeContracts: Invoking method 'GetValue' will always lead to an error. If this is wanted, consider adding Contract.Requires(false) to

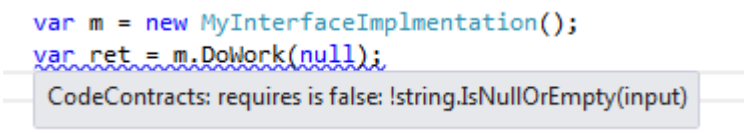
. . `ContractClassAttribute` ( ) `ContractClassForAttribute`

C # ...

```
[ContractClass(typeof(MyInterfaceContract))]
public interface IMyInterface
```

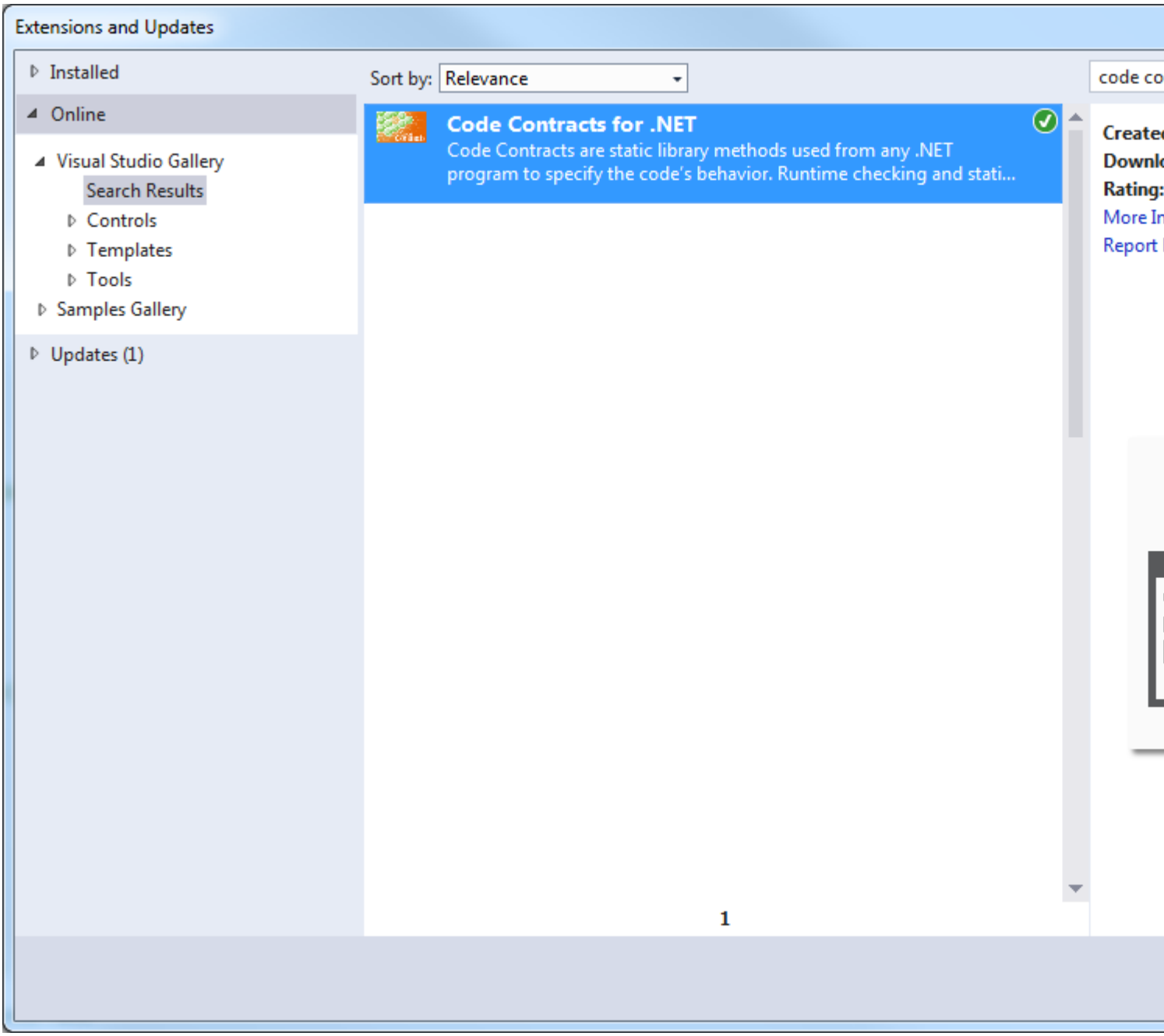
```
{
    string DoWork(string input);
}
//Never inherit from this contract defintion class
[ContractClassFor(typeof(IMyInterface))]
internal abstract class MyInterfaceContract : IMyInterface
{
    private MyInterfaceContract() { }

    public string DoWork(string input)
    {
        Contract.Requires(!string.IsNullOrEmpty(input));
        Contract.Ensures(!string.IsNullOrEmpty(Contract.Result<string>()));
        throw new NotSupportedException();
    }
}
public class MyInterfaceImplmentation : IMyInterface
{
    public string DoWork(string input)
    {
        return input;
    }
}
```

...

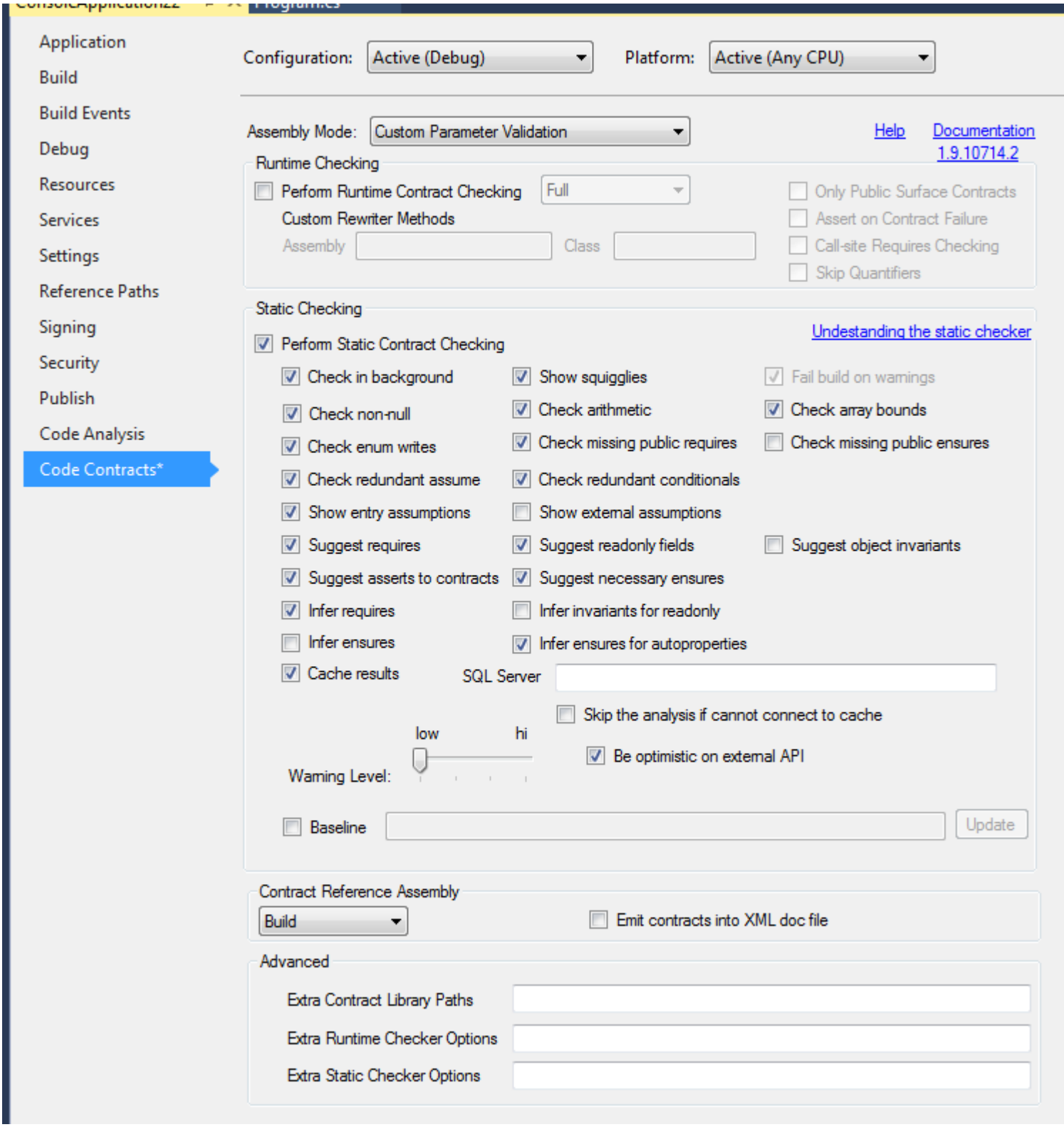


`System.Diagnostics.Contracts` .Net Framework . Visual Studio .

`Extensions and Updates` `Code Contracts` `Code Contracts Tools` .



`Code Contracts` . `Static Checking` (`Static Checking` . `Runtime Checking` .



: https://riptutorial.com/ko/dot-net/topic/1937/-

55: POST

Examples

WebRequest

/ .

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Net;
using System.Threading.Tasks;

public async Task<string> UploadFile(string url, string filename,
    Dictionary<string, object> postData)
{
    var request = WebRequest.CreateHttp(url);
    var boundary = $"{Guid.NewGuid():N}"; // boundary will separate each parameter
    request.ContentType = $"multipart/form-data; {nameof(boundary)}={boundary}";
    request.Method = "POST";

    using (var requestStream = request.GetRequestStream())
    using (var writer = new StreamWriter(requestStream))
    {
        foreach (var data in postData)
            await writer.WriteAsync( // put all POST data into request
                $"\r\n--{boundary}\r\nContent-Disposition: " +
                $"form-data; name=\"{data.Key}\"\r\n\r\n{data.Value}");

        await writer.WriteAsync( // file header
            $"\r\n--{boundary}\r\nContent-Disposition: " +
            $"form-data; name=\"File\"; filename=\"{Path.GetFileName(filename)}\"\r\n" +
            "Content-Type: application/octet-stream\r\n\r\n");

        await writer.FlushAsync();
        using (var fileStream = File.OpenRead(filename))
            await fileStream.CopyToAsync(requestStream);

        await writer.WriteAsync($"\r\n--{boundary}--\r\n");
    }

    using (var response = (HttpWebResponse) await request.GetResponseAsync())
    using (var responseStream = response.GetResponseStream())
    {
        if (responseStream == null)
            return string.Empty;
        using (var reader = new StreamReader(responseStream))
            return await reader.ReadToEndAsync();
    }
}
```

:

```
var response = await uploader.UploadFile("< YOUR URL >", "< PATH TO YOUR FILE >",
    new Dictionary<string, object>
    {
        {"Comment", "test"},
        {"Modified", DateTime.Now }
    });
```

POST : https://riptutorial.com/ko/dot-net/topic/10845/--post----

56: /

[redacted]

.( )

true false .

Examples

VB WriteAllText

```
Imports System.IO

Dim filename As String = "c:\path\to\file.txt"
File.WriteAllText(filename, "Text to write" & vbCrLf)
```

VB StreamWriter

```
Dim filename As String = "c:\path\to\file.txt"
If System.IO.File.Exists(filename) Then
    Dim writer As New System.IO.StreamWriter(filename)
    writer.Write("Text to write" & vbCrLf) 'Add a newline
    writer.close()
End If
```

C # StreamWriter

```
using System.Text;
using System.IO;

string filename = "c:\path\to\file.txt";
//'using' structure allows for proper disposal of stream.
using (StreamWriter writer = new StreamWriter(filename"))
{
    writer.WriteLine("Text to Write\n");
}
```

C # WriteAllText ()

```
using System.IO;
using System.Text;

string filename = "c:\path\to\file.txt";
File.writeAllText(filename, "Text to write\n");
```

C # File.Exists ()

```
using System;
using System.IO;

public class Program
{
    public static void Main()
    {
        string filePath = "somePath";

        if(File.Exists(filePath))
        {
            Console.WriteLine("Exists");
        }
        else
        {
            Console.WriteLine("Does not exist");
        }
    }
}
```

.

```
Console.WriteLine(File.Exists(pathToFile) ? "Exists" : "Does not exist");
```

/ : https://riptutorial.com/ko/dot-net/topic/1376/----

57:

.NET Framework . . Expression Trees C # . C # Expression <Func <... >> . LINQ . Entity Framework LINQ . Entity Framework SQL .

Examples

C

C # .

```
Expression<Func<int, int>> expression = a => a + 1;
```

C #

```
ParameterExpression parameterA = Expression.Parameter(typeof(int), "a");
var expression = (Expression<Func<int, int>>)Expression.Lambda(
                                                Expression.Add(
                                                    parameterA,
                                                    Expression.Constant(1)),
                                                parameterA);
```

. "a" 1 . CLR BinaryExpression Add of NodeType . . Left Right . Left "a" ParameterExpression Right 1 ConstantExpression.

.

```
Console.WriteLine(expression); //prints a => (a + 1)
```

C # .

C # CLR .

```
Func<int, int> lambda = expression.Compile();
Console.WriteLine(lambda(2)); //prints 3
```

SQL Reflection , .

==

```
_ => _.Field == "VALUE"
```

.

```
_ => _.Field   "VALUE",
```

.

- `IQueryable<T>` , `IEnumerable<T>` .
- `Linq`

to SQL Where .

Field "VALUE" Equal .

```
public static Expression<Func<T, bool>> BuildEqualPredicate<T>(
    Expression<Func<T, string>> memberAccessor,
    string term)
{
    var toString = Expression.Convert(Expression.Constant(term), typeof(string));
    Expression expression = Expression.Equal(memberAccessor.Body, toString);
    var predicate = Expression.Lambda<Func<T, bool>>(
        expression,
        memberAccessor.Parameters);
    return predicate;
}
```

Where .

```
var predicate = PredicateExtensions.BuildEqualPredicate<Entity>(
    _ => _.Field,
    "VALUE");
var results = context.Entity.Where(predicate).ToList();
```

.

```
public TestClass
{
    public static string StaticPublicField = "StaticPublicFieldValue";
}
```

StaticPublicField .

```
var fieldExpr = Expression.Field(null, typeof(TestClass), "StaticPublicField");
var labmda = Expression.Lambda<Func<string>>(fieldExpr);
```

.

```
Func<string> retriever = lambda.Compile();
var fieldValue = retriever();
```

// fieldValue StaticPublicFieldValue.

InvocationExpression

InvocationExpression Expression .

Expression.Invoke .

"" . null .

```
using System;
using System.Linq;
```

```
using System.Linq.Expressions;

public class Program
{
    public static void Main()
    {
        var elements = new[] {
            new Element { Description = "car" },
            new Element { Description = "cargo" },
            new Element { Description = "wheel" },
            new Element { Description = null },
            new Element { Description = "Madagascar" },
        };

        var elementIsInterestingExpression = CreateSearchPredicate(
            searchTerm: "car",
            whereToSearch: (Element e) => e.Description);

        Console.WriteLine(elementIsInterestingExpression.ToString());

        var elementIsInteresting = elementIsInterestingExpression.Compile();
        var interestingElements = elements.Where(elementIsInteresting);
        foreach (var e in interestingElements)
        {
            Console.WriteLine(e.Description);
        }

        var countExpensiveComputations = 0;
        Action incCount = () => countExpensiveComputations++;
        elements
            .Where(
                CreateSearchPredicate(
                    "car",
                    (Element e) => ExpensivelyComputed(
                        e, incCount
                    )
                ).Compile()
            )
            .Count();

        Console.WriteLine("Property extractor is called {0} times.",
countExpensiveComputations);
    }

    private class Element
    {
        public string Description { get; set; }
    }

    private static string ExpensivelyComputed(Element source, Action count)
    {
        count();
        return source.Description;
    }

    private static Expression<Func<T, bool>> CreateSearchPredicate<T>(
            string searchTerm,
            Expression<Func<T, string>> whereToSearch)
    {
        var extracted = Expression.Parameter(typeof(string), "extracted");
```

```
        Expression<Func<string, bool>> coalesceNullCheckWithSearch =
            Expression.Lambda<Func<string, bool>>(
                Expression.AndAlso(
                    Expression.Not(
                        Expression.Call(typeof(string), "IsNullOrEmpty", null, extracted)
                    ),
                    Expression.Call(extracted, "Contains", null,
Expression.Constant(searchTerm))
                ),
                extracted);

        var elementParameter = Expression.Parameter(typeof(T), "element");

        return Expression.Lambda<Func<T, bool>>(
            Expression.Invoke(
                coalesceNullCheckWithSearch,
                Expression.Invoke(whereToSearch, elementParameter)
            ),
            elementParameter
        );
    }
}
```

```
element => Invoke(extracted => (Not(IsNullOrEmpty(extracted)) AndAlso
extracted.Contains("car")), Invoke(e => e.Description, element))
car
cargo
Madagascar
Predicate is called 5 times.
```

Invoke propery .

```
Invoke(e => e.Description, element)
```

, e.Description , string extracted .

```
(Not(IsNullOrEmpty(extracted)) AndAlso extracted.Contains("car"))
```

AndAlso . 'false' . 'And' . NullReferenceException .

: https://riptutorial.com/ko/dot-net/topic/2657/-

58:

, 8 3 4 . 00001100 = 12

. , .

. 0 1 1 2 .

Examples

```
    public static int GetProcessAffinityMask(string processName = null)
    {
        Process myProcess = GetProcessByName(ref processName);

        int processorAffinity = (int)myProcess.ProcessorAffinity;
        Console.WriteLine("Process {0} Affinity Mask is : {1}", processName,
FormatAffinity(processorAffinity));

        return processorAffinity;
    }

    public static Process GetProcessByName(ref string processName)
    {
        Process myProcess;
        if (string.IsNullOrEmpty(processName))
        {
            myProcess = Process.GetCurrentProcess();
            processName = myProcess.ProcessName;
        }
        else
        {
            Process[] processList = Process.GetProcessesByName(processName);
            myProcess = processList[0];
        }
        return myProcess;
    }

    private static string FormatAffinity(int affinity)
    {
        return Convert.ToString(affinity, 2).PadLeft(Environment.ProcessorCount, '0');
    }
}
```

:

```
private static void Main(string[] args)
{
    GetProcessAffinityMask();

    Console.ReadKey();
}
// Output:
// Process Test.vshost Affinity Mask is : 11111111
```

```
    public static void SetProcessAffinityMask(int affinity, string processName = null)
    {
        Process myProcess = GetProcessByName(ref processName);

        Console.WriteLine("Process {0} Old Affinity Mask is : {1}", processName,
FormatAffinity((int)myProcess.ProcessorAffinity));

        myProcess.ProcessorAffinity = new IntPtr(affinity);
        Console.WriteLine("Process {0} New Affinity Mask is : {1}", processName,
FormatAffinity((int)myProcess.ProcessorAffinity));
    }
```

:

```
private static void Main(string[] args)
{
    int newAffinity = Convert.ToInt32("10101010", 2);
    SetProcessAffinityMask(newAffinity);

    Console.ReadKey();
}
// Output :
// Process Test.vshost Old Affinity Mask is : 11111111
// Process Test.vshost New Affinity Mask is : 10101010
```

: https://riptutorial.com/ko/dot-net/topic/4431/----

59:

- [DllImport ("Example.dll")] extern void SetText (string inString);
- [DllImport ("Example.dll")] extern GetText (StringBuilder outString);
- [MarshalAs (UnmanagedType.ByValTStr, SizeConst = 32)] ;
- [MarshalAs (UnmanagedType.ByValArray, SizeConst = 128)] byte [] byteArr;
- [StructLayout (LayoutKind.Sequential)] PERSON {...}
- [StructLayout (LayoutKind.Explicit)] MarshaledUnion {[FieldOffset (0)] ...}

Examples

Win32 dll

```
using System.Runtime.InteropServices;

class PInvokeExample
{
    [DllImport("user32.dll", CharSet = CharSet.Auto)]
    public static extern uint MessageBox(IntPtr hWnd, String text, String caption, int
options);

    public static void test()
    {
        MessageBox(IntPtr.Zero, "Hello!", "Message", 0);
    }
}
```

`static extern` stting `DllImportAttribute  Value` .dll name . `System.Runtime.InteropServices` . .

(Platform Invocation Services) .dll . P / Invoke .dll (, .NET Win32).

Windows API

pinvoke.net .

`extern` Windows API pinvoke.net . .

```
[DllImport("Example.dll")]
static extern void SetArray(
    [MarshalAs(UnmanagedType.LPArray, SizeConst = 128)]
    byte[] data);
```

```
[DllImport("Example.dll")]
static extern void SetStrArray(string[] textLines);
```

C ++ :

```
typedef struct _PERSON
{
```

```
    int age;
    char name[32];
} PERSON, *LP_PERSON;

void GetSpouse(PERSON person, LP_PERSON spouse);
```

C #

```
[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi)]
public struct PERSON
{
    public int age;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 32)]
    public string name;
}

[DllImport("family.dll", CharSet = CharSet.Auto)]
public static extern bool GetSpouse(PERSON person, ref PERSON spouse);
```

.

C ++

```
typedef struct
{
    int length;
    int *data;
} VECTOR;

void SetVector(VECTOR &vector);
```

`data` IntPtr `Marshal.AllocHGlobal()` `Marshal.FreeHGlobal()` `Marshal.FreeHGlobal()`).

```
[StructLayout(LayoutKind.Sequential)]
public struct VECTOR : IDisposable
{
    int length;
    IntPtr dataBuf;

    public int[] data
    {
        set
        {
            FreeDataBuf();
            if (value != null && value.Length > 0)
            {
                dataBuf = Marshal.AllocHGlobal(value.Length * Marshal.SizeOf(value[0]));
                Marshal.Copy(value, 0, dataBuf, value.Length);
                length = value.Length;
            }
        }
    }
    void FreeDataBuf()
    {
        if (dataBuf != IntPtr.Zero)
        {
            Marshal.FreeHGlobal(dataBuf);
```

```
            dataBuf = IntPtr.Zero;
        }
    }
    public void Dispose()
    {
        FreeDataBuf();
    }
}

[DllImport("vectors.dll")]
public static extern void SetVector([In]ref VECTOR vector);
```

.

C ++ :

```
typedef struct
{
    char *name;
} USER;

bool GetCurrentUser(USER *user);
```

IntPrt . Marshal.PtrToStringAnsi() .

```
[StructLayout(LayoutKind.Sequential)]
public struct USER
{
    IntPtr nameBuffer;
    public string name { get { return Marshal.PtrToStringAnsi(nameBuffer); } }
}

[DllImport("users.dll")]
public static extern bool GetCurrentUser(out USER user);
```

C ++

```
typedef union
{
    char c;
    int i;
} CharOrInt;
```

C #

```
[StructLayout(LayoutKind.Explicit)]
public struct CharOrInt
{
    [FieldOffset(0)]
    public byte c;
    [FieldOffset(0)]
    public int i;
}
```

~~FieldOffset(0) text; FieldOffset(0) i;~~ ~ .

```
typedef union
{
    char text[128];
    int i;
} TextOrInt;
```

. .

```
[StructLayout(LayoutKind.Sequential)]
public struct TextOrInt
{
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 128)]
    public byte[] text;
    public int i { get { return BitConverter.ToInt32(text, 0); } }
}
```

: https://riptutorial.com/ko/dot-net/topic/1643/-

S. No		Contributors
1	.NET Framework	Adriano Repetti, Alan McBee, ale10ander, Andrew Jens, Andrew Morton, Andrey Shchekin, Community, Daniel A. White, Ehsan Sajjad, harriyott, hillary.fraley, Ian, James Thorpe, Jamie Rees, Joel Martinez, Kevin Montrose, Lirrik, MarcinJuraszek, matteeyah, naveen, Nicholas Sizer, Paweł Izdebski, Peter, Peter Gordon, Peter Hommel, PSN, Richard Lander, Rion Williams, Robert Columbia, RubberDuck, SeeuD1, Serg Rogovtsev, Squidward, Stephen Leppik, Steven Doggart, svick, ɬolɐəz əuɬ qoq
2	.NET	Mihail Stancescu
3	.NET JSON Newtonsoft.Json	DLeh
4	ADO.NET	Akshay Anand, Andrew Morton, Daniel A. White, DavidG, Drew, elmer007, Hamid, Harjot, Heinzi, Igor, user2321864
5	ASP.NET ASP.NET MVC	Scott Hannen
6	C # SHA1	mahdi abasi
7	CLR	Gajendra, starbeamrainbowlabs, Theodoros Chatzigiannakis
8	DateTime	GalacticCowboy, John
9	HTTP	Devon Burriss, Konamiman
10	HTTP	CodeCaster, Konamiman, MuiBienCarlota
11	JIT	Krikor Ailanjian
12	JSON	Akshay Anand, Andrius, Eric, hasan, M22an, PedroSouki, Thriggle, Tolga Evcimen
13	LINQ	A. Raza, Adil Mammadov, Akshay Anand, Alexander V., Benjamin Hodgson, Blachshma, Bradley Grainger, Bruno Garcia, Carlos Muñoz, CodeCaster, dbasnett, DoNot

		, dotctor, Eduardo Molteni, Ehsan Sajjad, GalacticCowboy, H. Pauwelyn, Haney, J3soon, jbtule, jnovo, Joe Amenta, Kilazur, Konamiman, MarcinJuraszek, Mark Hurd, McKay, Mellow, Mert Gülsoy, Mike Stortz, Mr.Mindor, Nate Barbettini, Pavel Voronin, Ruben Steins, Salvador Rubio Martinez, Sammi, Sergio Domínguez, Sidewinder94
14	NuGet	Andrey Shchekin, Anik Saha, Ashtonian, CodeCaster, Daniel A. White, Matas Vaitkevicius, Ozair Kafray
15	ReadOnlyCollections	tehDorf
16	StdErr	Aleks Andreev
17	System.IO	CodeCaster, Daniel A. White, demonplus, Filip Frącz, RoyalPotato
18	System.IO.File	Adriano Repetti, delete me
19	System.Net.Mail	demonplus, Steve, vicky
20	System.Reflection.Emit	Luaan, NikolayKondratyev, RamenChef, toddmo
21	System.Runtime.Caching.MemoryCache (ObjectCache)	Guanxi, RamenChef
22	TPL	i3arnon, Jacobr365, Nikola.Lukovic, RamenChef
23	VB	ale10ander, dbasnett
24	XmlSerializer	Aphelion, George Polevoy, RamenChef, Rowland Shaw, Thomas Levesque, void, Yogi
25	Zip	Arxae
26		Dr Rob Lang, just.ru, Lucas Trzesniewski
27		Joe Amenta, Kirk Broadhurst, RamenChef
28		Konamiman
29		Axarydax
30		Yahfoufi

31		DLeh, Gusdor
32		Big Fan, binki, DrewJordan
33		Adriano Repetti, Alexander Mandt, Matt, Pavel Voronin, RamenChef
34		Aleks Andreev, Bjørn-Roger Kringsjå, demonplus, Jean-Baptiste Noblot, Jigar, JJP , Kirk Broadhurst, Lorenzo Dematté, Matas Vaitkevicius, NetSquirrel, Pavel Mayorov, Peter, smdrager, Terry, user1304444, void
35		Alan McBee, DrewJordan, matteeyah
36		Adriano Repetti, Bjørn-Roger Kringsjå, Daniel Plaisted, Darrel Lee, Felipe Oriani, George Duckett, George Polevoy, hatchet, Hogan, Ian, LegionMammal978, Luke Bearl, Olivier Jacot-Descombes, RamenChef, Ringil, Robert Columbia, Stephen Byrne, the berserker, Tomáš Hübelbauer
37		Alan McBee
38		Behzad, Martijn Pieters, Mellow
39		Hywel Rees
40	SpeechRecognitionEngine	ProgramFOX, RamenChef
41		Adi Lester, Bassie, Fredou, Ogglas, Ondřej Štorc, RamenChef
42		avat
43	/	Alexander Mandt, Daniel A. White, demonplus, Jagadisha B S, lokusking, Matt
44		Adi Lester, Akshay Anand, Alan McBee, Alfred Myers, Arvin Baccay, BananaSft, CodeCaster, Dave R., Kritner, Mafii, Matt, Rob, Sean, starbeamrainbowlabs, STW, Yousef Al-Mulla
45		Tanveer Badar
46		Phil Thomas, Scott Hannen
47	(TPL)	Adi Lester, Aman Sharma, Andrew, i3arnon,

		Jacobr365, JamyRyals, Konamiman, Mathias Müller, Mert Gülsoy, Mikhail Filimonov, Pavel Mayorov, Pavel Voronin, RamenChef, Thomas Bledsoe, TorbenJ
48	(TPL) API	Gusdor, Jacobr365
49	(System.Text.RegularExpressions)	BrunoLM, Denuath, Matt dc, tehDorf
50		Dmitry Egorov
51	IProgress	DLeh
52		Alan McBee, Aman Sharma, Anik Saha, Daniel A. White, demonplus, Felipe Oriani, harriyott, Ian, Mark C., Ravi A., Virtlink
53		JJS, Matthew Whited, RamenChef
54	POST	Aleks Andreev
55	/	ale10ander, Alexander Mandt, Ingenioushax, Nitram
56		Akshay Anand, George Polevoy, Jim, n.podbielski, Pavel Mayorov, RamenChef, Stephen Leppik, Stilgar, wangengzheng
57		MSE, RamenChef
58		Dmitry Egorov, Imran Ali Khan