netgenium

Programming of print templates in PDF format

Framework NET Genium

netgenium.com

Content

1 Subject and purpose of the manual

This guide attempts to describe all the resources needed to create various print templates in PDF format. To create print templates, you need:

- Visual Studio 2013 with Report Designer
- "NETGeniumConnection.dll" and "O2S.Components.PDF4NET.dll" libraries
- NET Genium installed on the local computer
- External functions project "ngef"
- Application for viewing "pdf" files

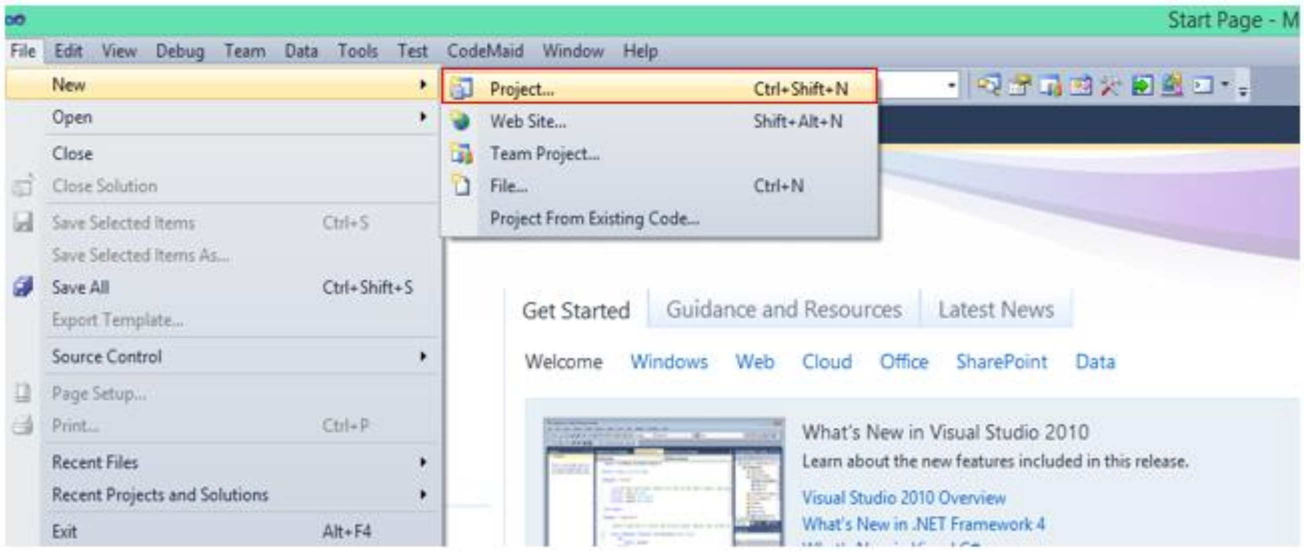
The required output is the incorporation of a new print template in the form of C# code into an external function. To do this, you must first design a new print template as part of a new project – such as a console application – in the Visual Studio developer tool. Then run this application, which creates C# source code that can be pasted into an external function.

Simplified procedure:

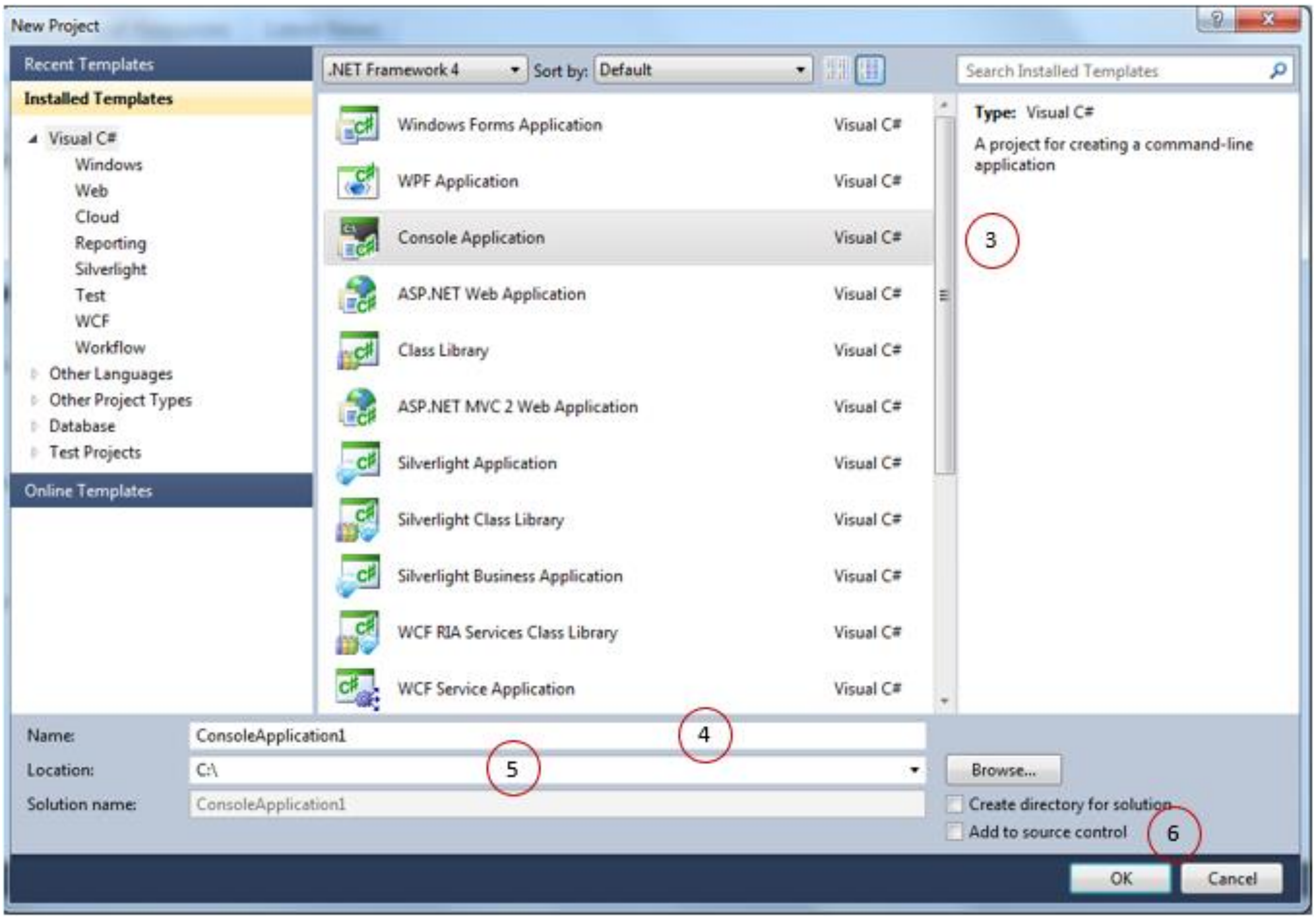
1. Install Visual Studio 2013 (or higher)
2. Create a console application
3. Add references to the required libraries to run the console application
4. Paste the code into the Program class
5. Create an "rdlc" file
6. Run the console application + view the generated files
7. Use the individual Visual Studio tools to draw the template
8. Set the properties of each element
9. Run the final version of the console application
10. Incorporate a print template into an external function

2 Creating a console application

- Start the Visual Studio program (start the shortcut from the desktop or via the menu “Start” – “Programs” – “Microsoft Visual Studio” – “Microsoft Visual Studio”).
- Select “File” – “New” – “Project” from the menu.



- Select “Console Application” as the project.
- Name the new project.
- Enter the location of the project.
- Press the “OK” button.



New Project

Name: ConsoleApplication1

Location: C:\

Solution name: ConsoleApplication1

Type: Visual C#

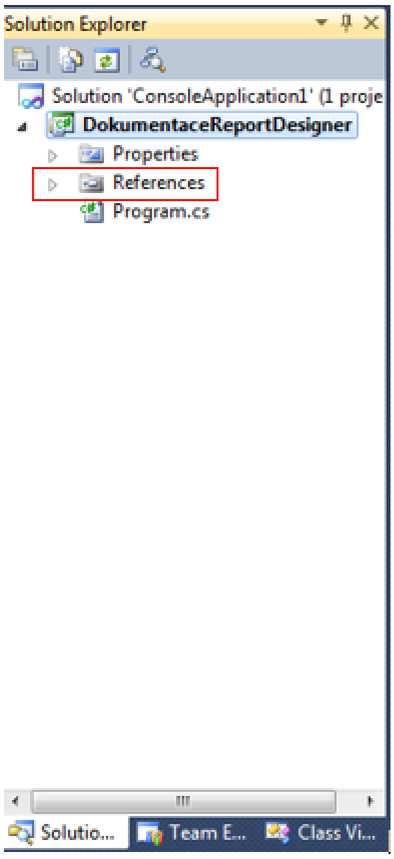
A project for creating a command-line application

Recent Templates
Installed Templates
Visual C#
Windows
Web
Cloud
Reporting
Silverlight
Test
WCF
Workflow
Other Languages
Other Project Types
Database
Test Projects
Online Templates
.NET Framework 4
Sort by: Default
Search Installed Templates
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
ASP.NET Web Application Visual C#
Class Library Visual C#
ASP.NET MVC 2 Web Application Visual C#
Silverlight Application Visual C#
Silverlight Class Library Visual C#
Silverlight Business Application Visual C#
WCF RIA Services Class Library Visual C#
WCF Service Application Visual C#
3
4
5
6
Browse...
Create directory for solution
Add to source control
OK
Cancel

3 Add references

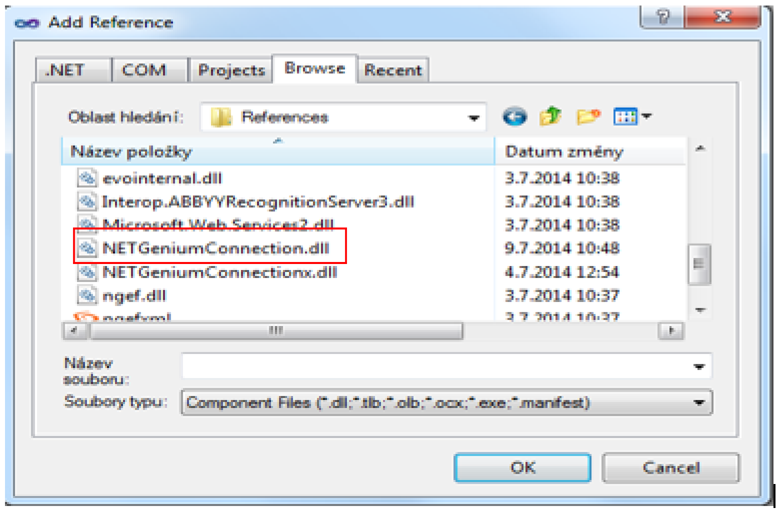
To run the console application correctly, it is important to add references to the libraries in the project, which are stored in the "N:\NetGenium\Projects\NetGenium\References" directory. This is done as follows:

Right-click on the "References" folder in the "Solution Explorer" window.



Select the "Browse" tab and navigate to the "N:\NetGenium\Projects\NetGenium\References" directory.

- Select the “NETGeniumConnection.dll” file from the “dll” files provided.



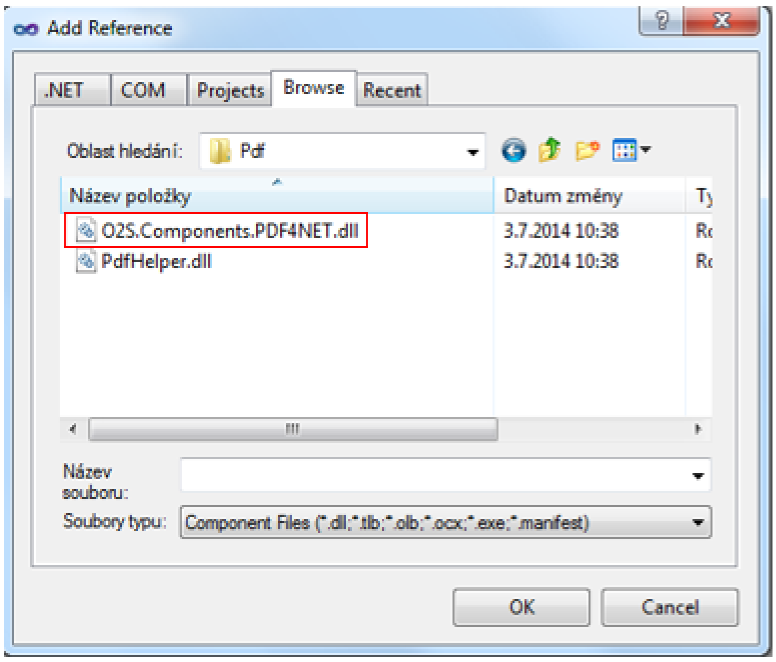
Oblast hledání: References

Název položky	Datum změny
evointernal.dll	3.7.2014 10:38
Interop.ABBYYRecognitionServer3.dll	3.7.2014 10:38
Microsoft.Web.Services2.dll	3.7.2014 10:38
NETGeniumConnection.dll	9.7.2014 10:48
NETGeniumConnectionx.dll	4.7.2014 12:54
ngef.dll	3.7.2014 10:37
ngef.xml	3.7.2014 10:37

Název souboru:

Soubory typu: Component Files (*.dll;*.tlb;*.olb;*.ocx;*.exe;*.manifest)

- In the “N:\NetGenium\Projects\NetGenium\References\Pdf” directory, select the “O2S.Components.PDF4NET.dll” file.



Oblast hledání: Pdf

Název položky	Datum změny	Ty
O2S.Components.PDF4NET.dll	3.7.2014 10:38	Ro
PdfHelper.dll	3.7.2014 10:38	Ro

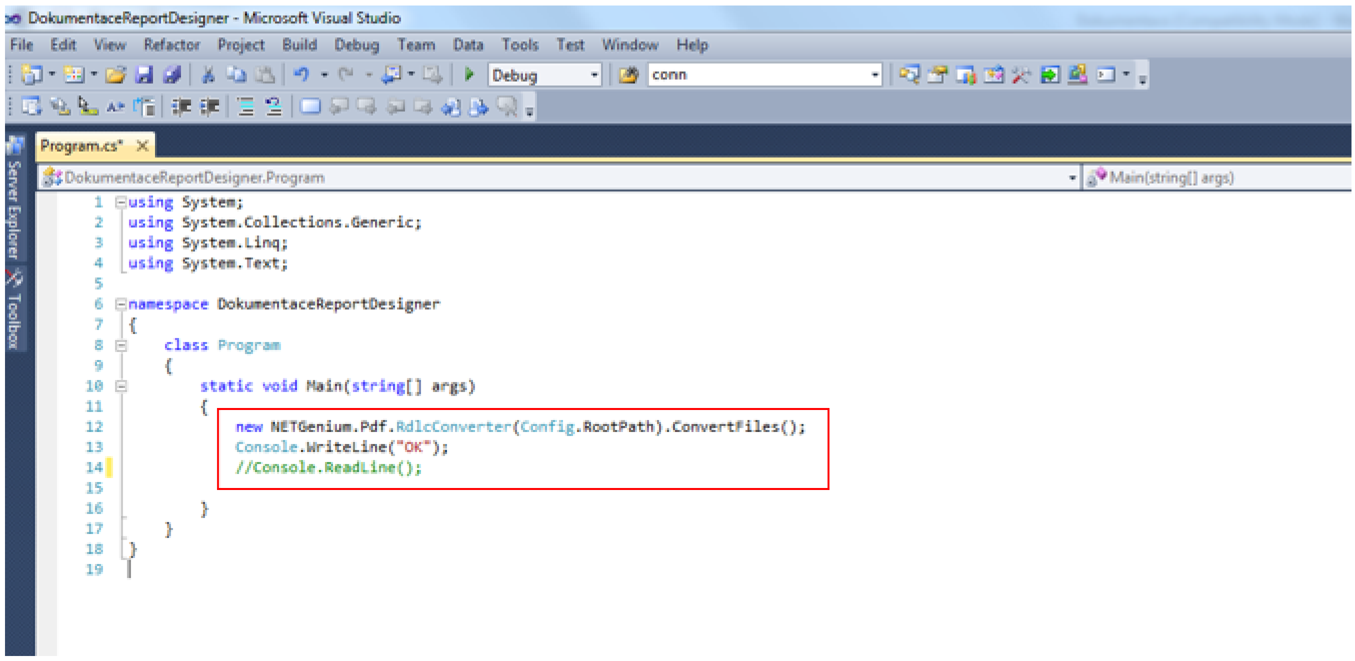
Název souboru:

Soubory typu: Component Files (*.dll;*.tlb;*.olb;*.ocx;*.exe;*.manifest)

4 Writing code in the Program class

After creating a new console application, Visual Studio automatically created a new “Program” class, as shown in the image below. Method “Main(string[] args)” of the “Program” class is now empty, so in the next step, insert the corresponding code into the body of this method. This code consists of the following 3 lines:

```
new NETGenium.Pdf.RdlcConverter(Config.RootPath).ConvertFiles();
Console.WriteLine("OK");
// Console.ReadLine();
```


DokumentaceReportDesigner - Microsoft Visual Studio
File Edit View Refactor Project Build Debug Team Data Tools Test Window Help
Debug
conn
Server Explorer
Toolbox
Program.cs*
DokumentaceReportDesigner.Program
Main(string[] args)

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

namespace DokumentaceReportDesigner
{
    class Program
    {
        static void Main(string[] args)
        {
            new NETGenium.Pdf.RdlcConverter(Config.RootPath).ConvertFiles();
            Console.WriteLine("OK");
            //Console.ReadLine();

        }
    }
}
```

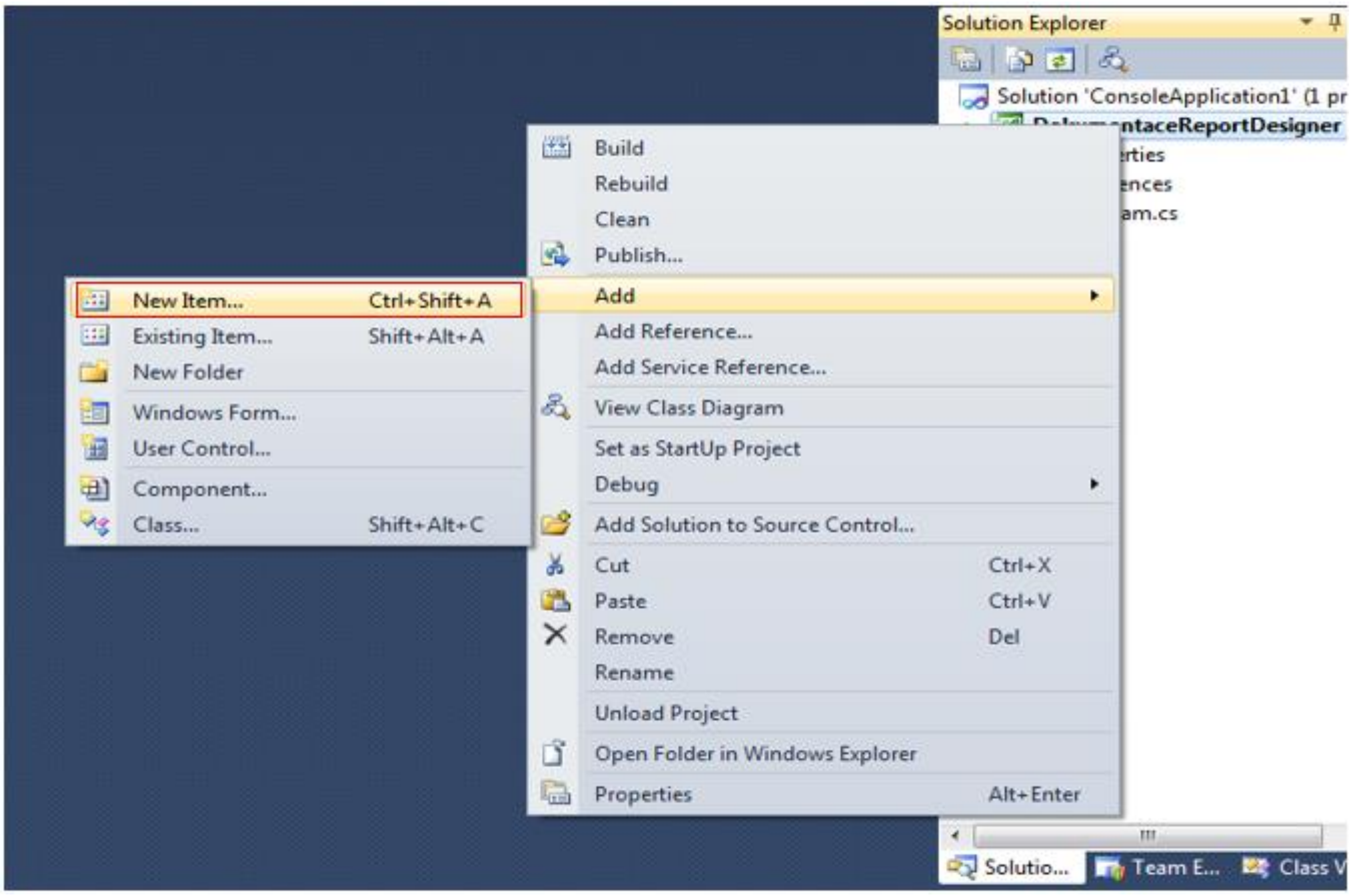
The method, which is on the first inserted line, searches for all “rdlc” files that exist in the project, and then creates files with the extension:

- “cs” – the resulting source code, which we will incorporate into an external function later after tuning the print template.
- “pdf” – a preview of the “pdf” of the document, which is created by running the source code above – is used for checking during debugging the print template.

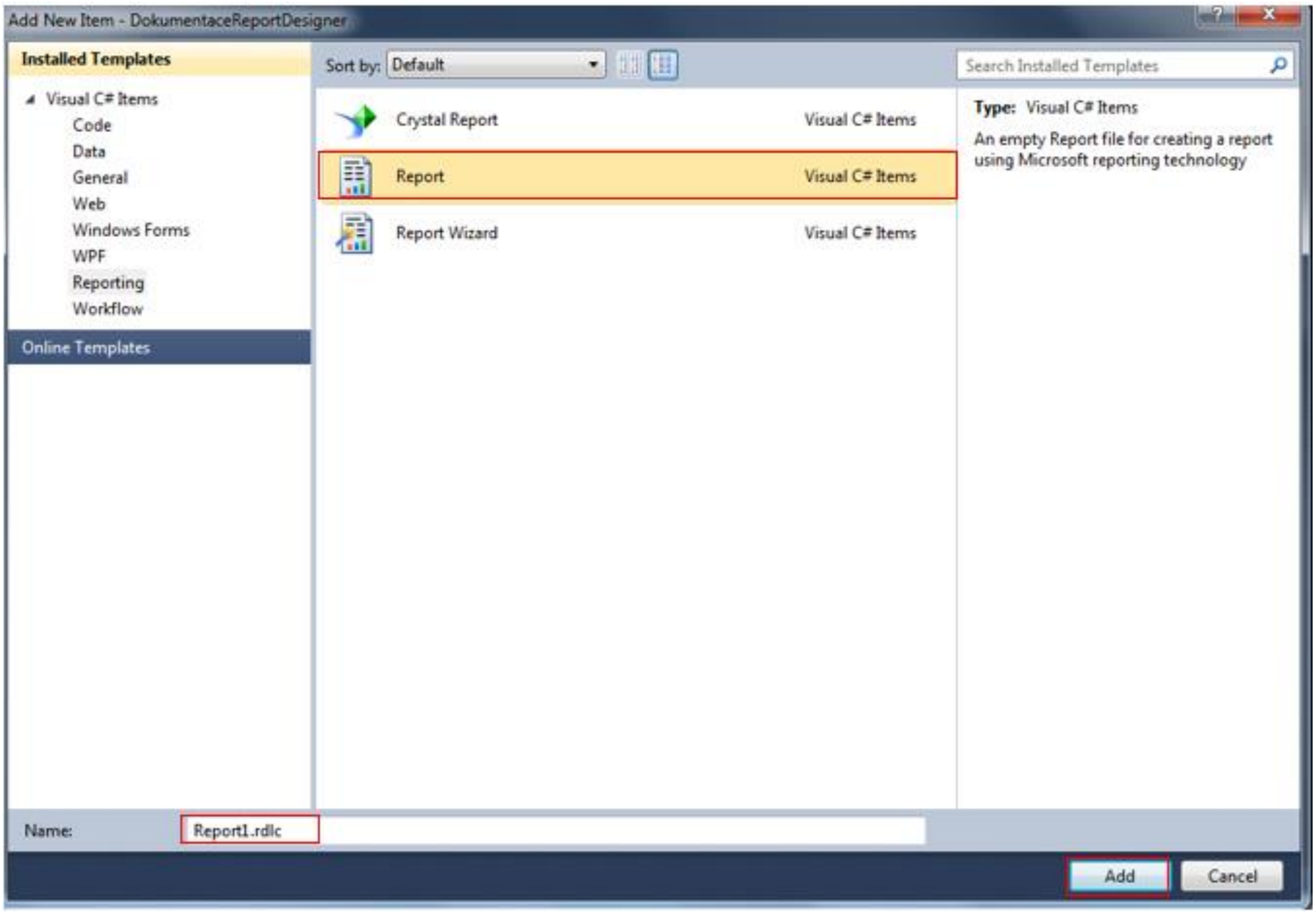
5 Creating an “rdlc” file

The print template design itself will be created in a special file. This file has an “rdlc” extension. This file is inserted into the project you are creating as follows:

- Right-click on the project name in the “Solution Explorer” window.
- Select “Add” – “New Item” from the menu.



- Select the file to be created – select “Reporting” from the “Installed Templates” menu and select “Report” from these reporting templates. You can also set the name of this file (it is important that the file has the extension “rdlc”) and confirm your selection by clicking on the “Add” button.

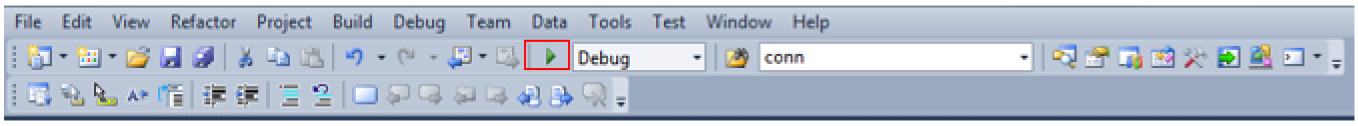

Add New Item - DokumentaceReportDesigner
Installed Templates
Visual C# Items
Code
Data
General
Web
Windows Forms
WPF
Reporting
Workflow
Online Templates
Sort by: Default
Search Installed Templates
Crystal Report Visual C# Items
Report Visual C# Items
Report Wizard Visual C# Items
Type: Visual C# Items
An empty Report file for creating a report using Microsoft reporting technology
Name: Report1.rdlc
Add
Cancel

- Clicking the “Add” button opens the “rdlc” file created in the previous steps in “Design” mode. Another possible mode is the “XML” mode, which can be opened by right-clicking on the “rdlc” file, selecting “Open With…” from the menu, and selecting “XML (Text) Editor” from the newly opened menu.

- *However, the most important mode for creating a print template will be the “Design” mode, in which the next workflow will be described and on which the mentioned “XML” is based.*

6 Start the console application

- It is now possible to run a console application to check that the written code is correct and to insert all the necessary libraries. To do this, press the green arrow in the top panel or press the "F5" key.

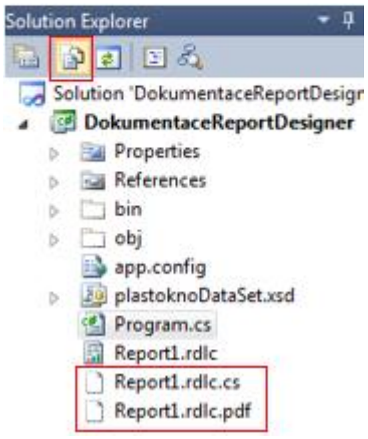

File Edit View Refactor Project Build Debug Team Data Tools Test Window Help
Debug
conn

After starting the console application, the already mentioned pair of files ("cs" and "pdf") will be created. These files will be created in the same folder as the "rdlc" file. This means that it is possible to open them, for example, using the Total Commander program. In addition, both files can be opened and seen in the project structure in Visual Studio.

Procedure in Visual Studio:

- Open the "Solution Explorer" window.
- Press the "Show all files" button.

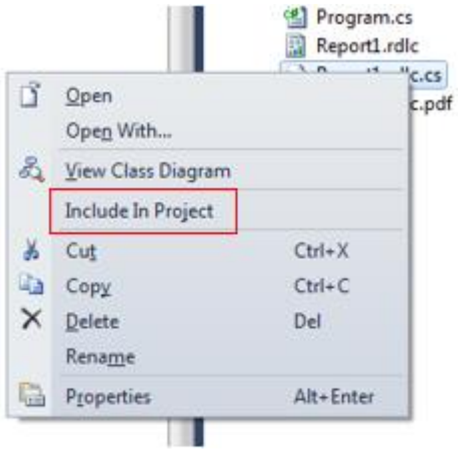
All files in the project are displayed.



These files are always overwritten after each run, so if you need to keep the current C# code file, you must save the file under a different name than the original file.

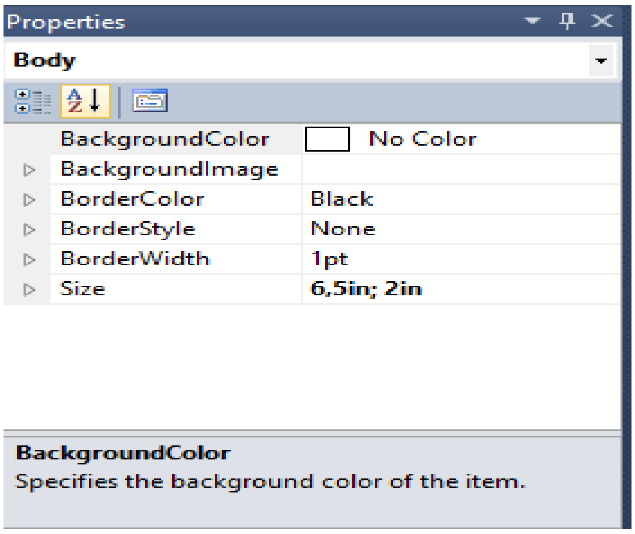
The most important of these two files is the one with the "cs" extension. In most cases, this file will be further modified so that it can be inserted into some external function. Before editing, it is possible to check the file that was created automatically by running it for any errors (references, etc.).

- In the “Solution Explorer” window, right-click on the “cs” file and select “Include In Project”. The file can be excluded from the project in the same way. Except that you do not use “Include” but “Exclude”.



7 Visual Studio tools for creating a report

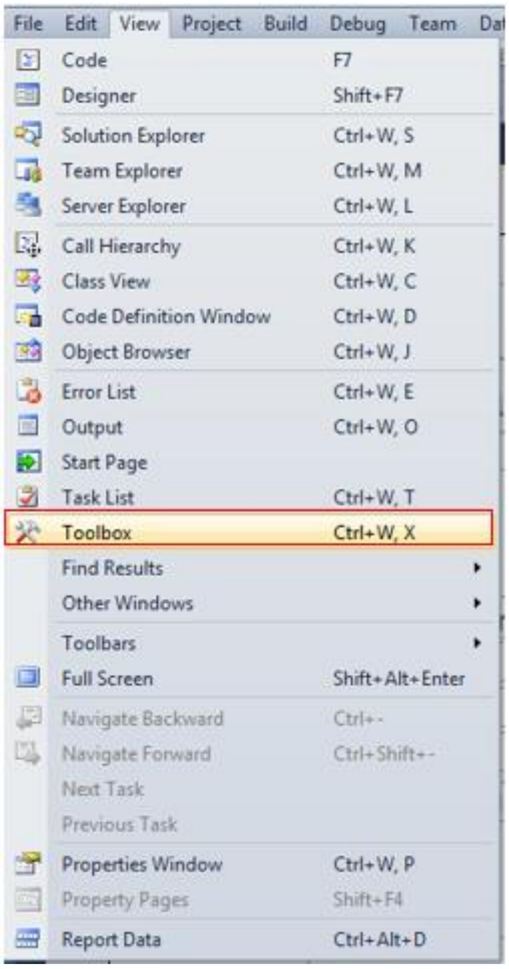
Visual Studio offers a fairly large number of ways to create any print template in the “rdlc” file. When you open the “rdlc” file in “Design” mode, there is a drawing canvas on the main screen on which the future print template will be drawn. After clicking with the left button in the body of the drawing canvas, the “Properties” window is available on the right side, in which the properties of the selected element can be set. In this case, it is the “Body” element. The “Properties” window is also available for all other reporting elements that Visual Studio offers, so it is possible to set various properties for lines, text boxes, rectangles, tables and images. The “Properties” window can also be opened by clicking on the “View” tab and selecting the “Properties window” option.



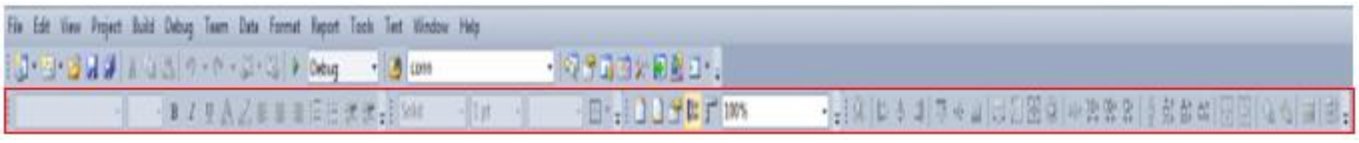
BackgroundColor	No Color
BackgroundImage	
BorderColor	Black
BorderStyle	None
BorderWidth	1pt
Size	6,5in; 2in

BackgroundColor
Specifies the background color of the item.

- *Elements can be inserted into the canvas using the "Toolbox", which is available on the left or in the top menu under the "View" tab and the "Toolbox" option.*

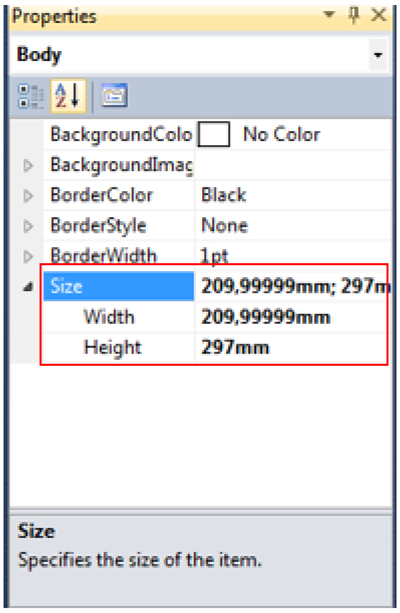


- *Another way to set the properties of individual elements is the line, which is located below the top menu and is very similar to the setting options available in Word, for example, so the user can use this option if he does not like the settings via the "Properties" window.*



8 Options for setting individual elements

- First of all, set the paper parameters when creating a new print template. These parameters are set in the "Points" element. Left-click anywhere on the artboard and set the height and width of the artboard ("Size" parameter) in the "Properties" window. The figure below shows the parameters for size A4 (210 mm x 297 mm). When you set the width to 210 mm, Visual Studio resets the value to the new value of 209.99999 mm.



Properties

Body

BackgroundColo	No Color
BackgroundImag	
BorderColor	Black
BorderStyle	None
BorderWidth	1pt
Size	209,99999mm; 297m
Width	209,99999mm
Height	297mm

Size
Specifies the size of the item.

After setting the dimensions of the drawing area, it is possible to insert individual elements into it, while it is advisable to check the setting of the drawing area dimensions at the end of creating a print template, because area and to increase the dimensions of that area.
Elements that will be inserted into the print template include:

- Text Box
- Line
- Image
- Rectangle
- Table

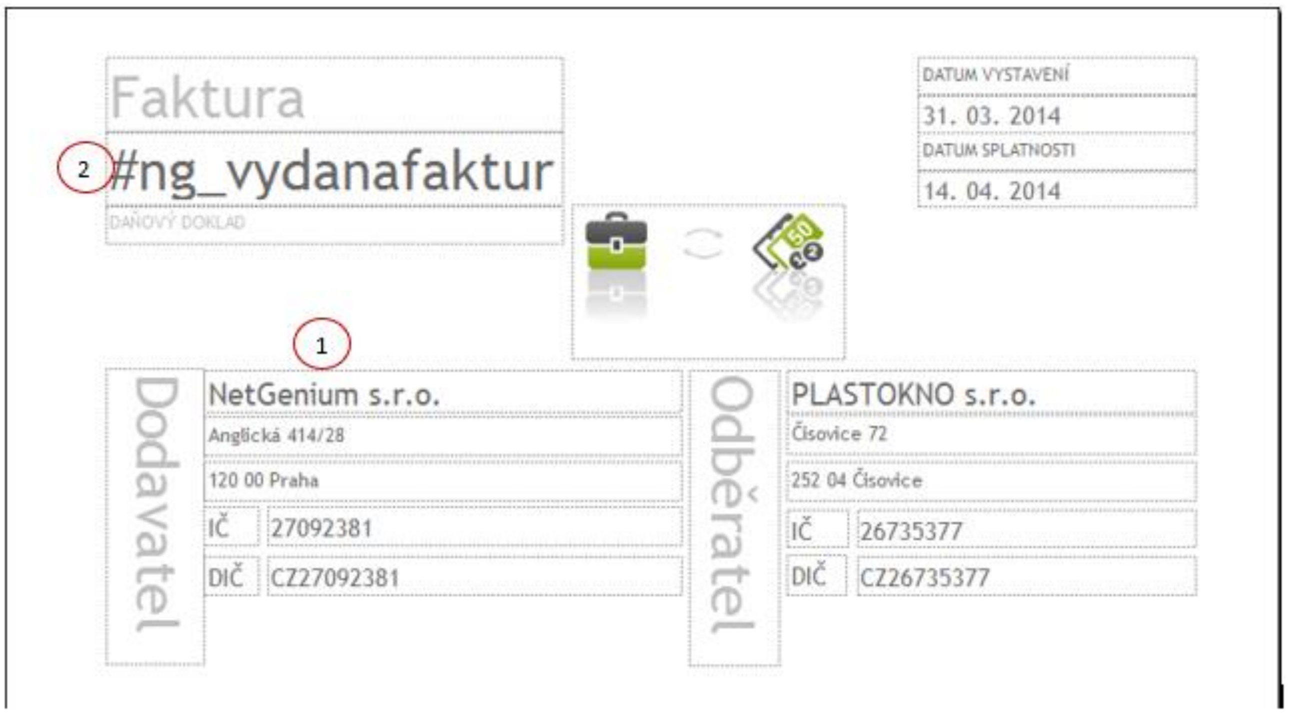
All these elements can be found in the "Toolbox" and dragged to the drawing area.

8.1 Text Box

After dragging to the drawing area, it is possible to set various properties in the "Properties" window for individual text fields. These features are:

Property	Importance
BackgroundColor	Set background color print template
BorderColor	Text box border color settings
BorderStyle	Sets the type of line that borders the textbox. Options: "Default" "Dashed", "Dotted", "Double", "None", "Solid"
BorderWidth	Adjust the thickness of the line delimiting the textbox
Color	Font color settings
FontFamily	Font settings
FontSize	Font size setting
FontStyle	Font style settings. Italic style setting option
FontWeight	Font thickness setting. Option to set "Bold".
PaddingLeft	Left padding settings
PaddingRight	Right padding settings
PaddingTop	Top padding settings
PaddingBottom	Lower padding settings
TextAlign	Adjust horizontal alignment. Setting options: "Default" "General", "Left", "Center", "Right"
VerticalAlign	Vertical alignment adjustment. Setting options: "Top", "Middle", "Bottom"
WritingMode	Vertical/horizontal text box settings. Option "Vertical", "Horizontal"

Further in the "Properties" window are parameters such as "Location", "Size" and "Value". However, it is not necessary to set these parameters in the "Properties" window, because they reset themselves when the user changes the position/size/text on the canvas.
The text itself inside the textbox can be of two types – static and dynamic. Under the word static, you can imagine text that will be inserted into the print template as a constant. This means that it will not change after generating a print template in NET Genium. The second type of text is the so-called dynamic. It will be read from the database, and therefore will be different for each print template.
The difference between the two types of text is evident in the figure below, because the static text is written in normal text (example 1 in the figure: "NET Genium s.r.o."), while the dynamic text is expressed as follows "#ng_tablename#.#ng_columnname#" (example 2 in the figure: "ng_fv#.#ng_streetnumber#"). This label allows you to distinguish between the two types of text at runtime and then project the difference into C# code.



8.2 Line

The Line element appears in Report Designer as a line that can be set with the following parameters.

Property	Importance
LineColor	Line color settings
LineStyle	Line type setting. Options: “Dashed”, “Dotted”, “Solid”
LineWidth	Line thickness setting

Other parameters, such as the position and length of the line, can be set, as in the case of the Textbox element, by dragging on the drawing canvas.

8.3 Image

- *You can also insert a picture into the artboard, which will then be in the print template. The image is inserted into the artboard as "Embedded", which means that the image must be imported into the project. To insert the image correctly, you must first select the "Image" element from the "Toolbox". After dragging this element to the artboard, the "Image Properties" window opens. Here you need to set the data source ("Select the image source:") to "Embedded" and then press the "Import" button, which can be used to specify the path to the selected image. After importing the image to the artboard, the "OK" button is pressed and the imported image can be seen on the artboard.*

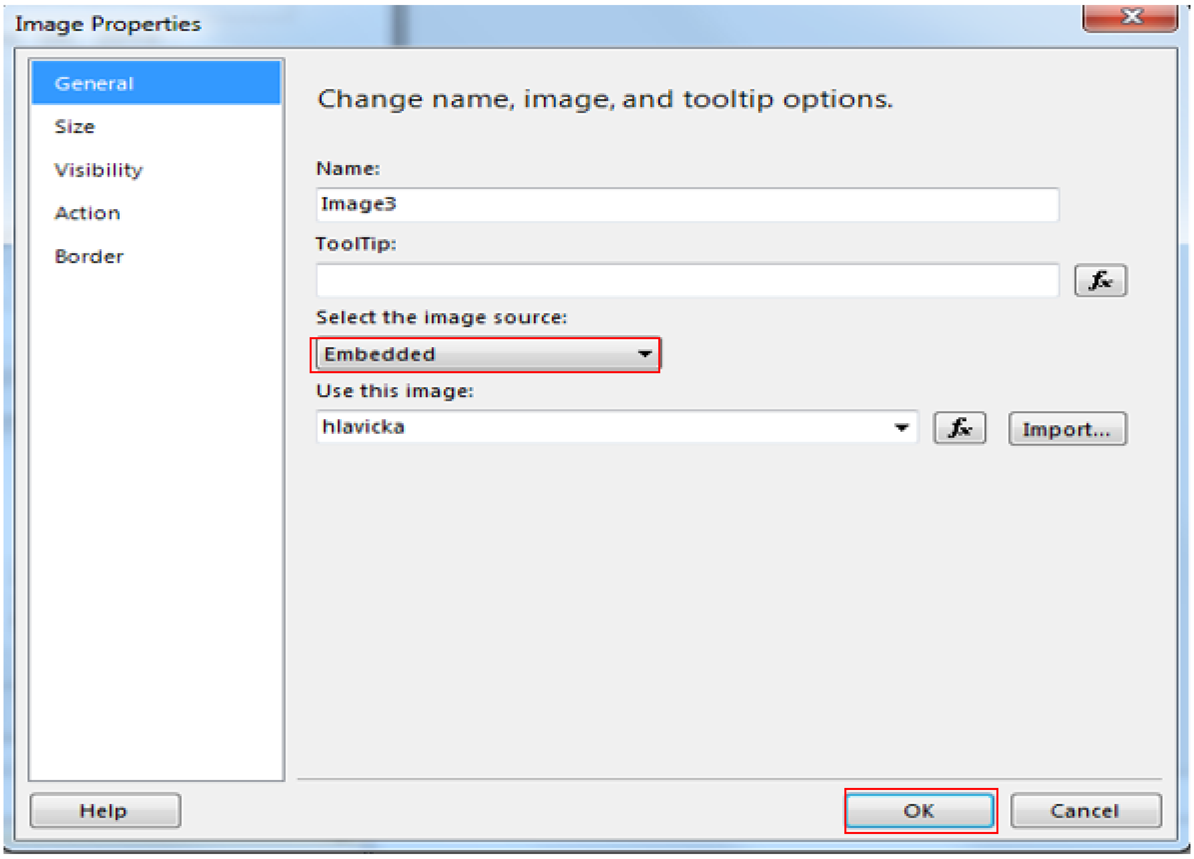


Image Properties

Change name, image, and tooltip options.

Name: Image3

ToolTip:

Select the image source: Embedded

Use this image: hlavicka

- *All parameters that can be specified for the Image element can be edited directly on the artboard. In most cases, this is the position of the image and its height and width.*

8.4 Rectangle

For a rectangle element, parameters such as "Location", the height and width of the rectangle can be determined again directly on the canvas by dragging the element across the canvas. However, the rectangle still offers additional settings that must already be specified in the "Properties" window. These features are:

Property	Importance
BackgroundColor	Frame background color settings
BorderColor	Adjust the border color of the frame
BorderStyle	Border line type setting. Options: "Default", "Dashed", "Dotted", "Double", "None", "Solid"
BorderWidth	Border thickness setting

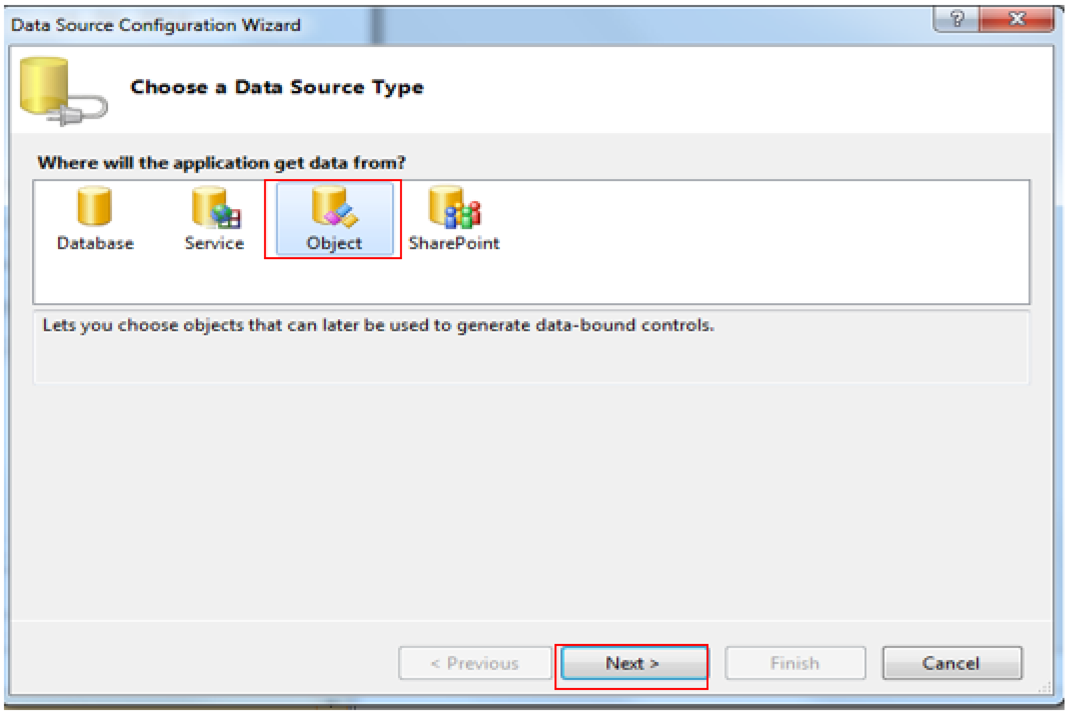
8.5 Table

The table is the last element needed to create print templates. The Table element must have a data source specified because Visual Studio reports an error when starting a console application and not defining a data source. After dragging the Table element from the "Toolbox" to the canvas, the "Tablix Properties" window opens and, if no data source has been defined before, a window opens in which the data source should be identified. For this purpose, there is a wizard that will set up the data source. If the data source were not defined, Visual Studio would allow the Table element to be inserted on the canvas, but the "Debugger" would report an error the next time it was run.

Steps to set up a data source:

1. Select what type the data source will be.

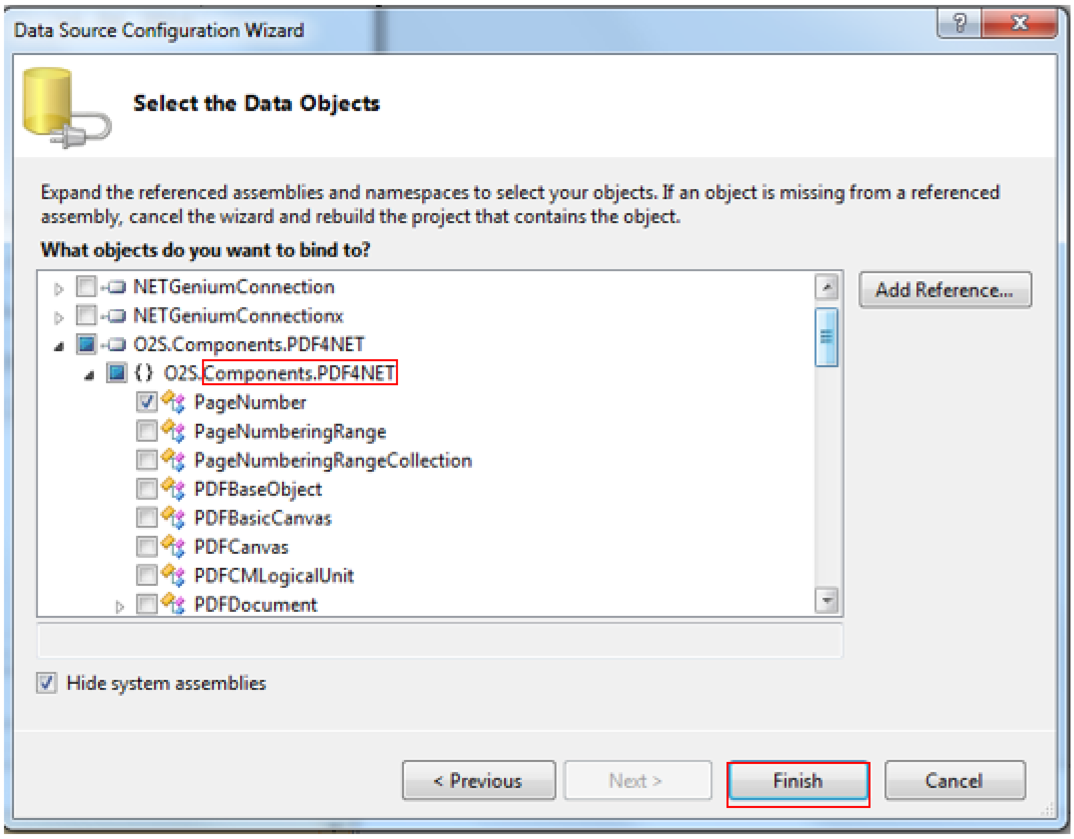
☞ Select “Object” and press “Next”.


Data Source Configuration Wizard
Choose a Data Source Type
Where will the application get data from?
Database
Service
Object
SharePoint
< Previous
Next >
Finish
Cancel

Lets you choose objects that can later be used to generate data-bound controls.

2. Select a library from the options offered.

Select "O2S.Components.PDF4NET/O2S.Components.PDF4NET/PageNumber" and press the "Finish" button.


Data Source Configuration Wizard

Select the Data Objects

Expand the referenced assemblies and namespaces to select your objects. If an object is missing from a referenced assembly, cancel the wizard and rebuild the project that contains the object.

What objects do you want to bind to?

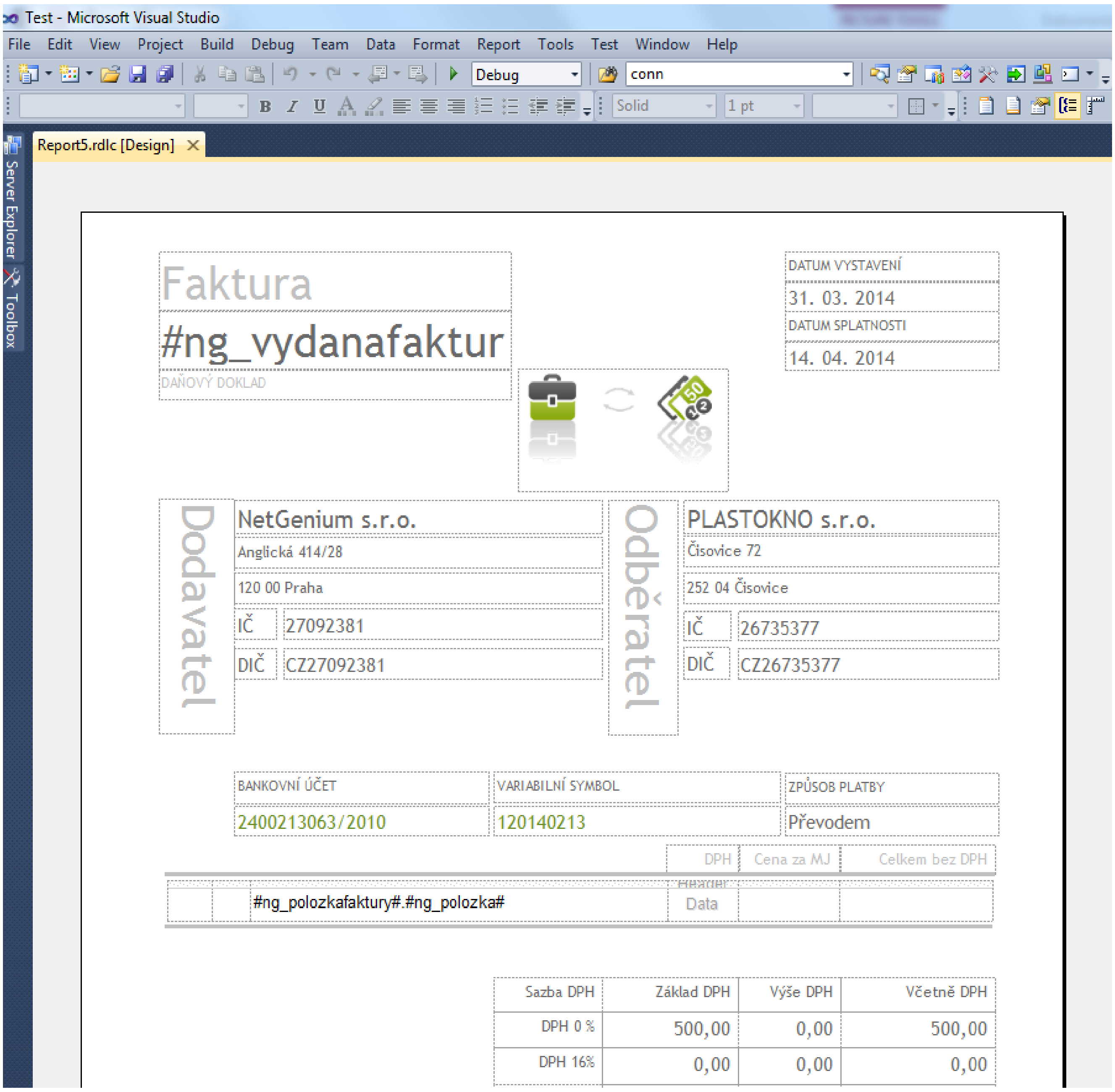
- NETGeniumConnection
- NETGeniumConnectionx
- O2S.Components.PDF4NET
  - O2S.Components.PDF4NET
    - [✓] PageNumber
    - PageNumberingRange
    - PageNumberingRangeCollection
    - PDFBaseObject
    - PDFBasicCanvas
    - PDFCanvas
    - PDFCMLogicalUnit
    - PDFDocument

[✓] Hide system assemblies

Table parameters that are important to a print template can already be set on the artboard by dragging an element or resizing. On the canvas it is possible to set such parameters as "Location" (table position), width of individual columns, height of individual rows, texts in the table, number of columns, number of rows.
The individual cells of the table are text boxes, so it is possible to set the same parameters as in the case of classic text boxes. The table element is mainly used for items. In C# code, items are inserted into a print template using a loop, because in most cases there are more of them in the database. To ensure that the cycle is written in C# code after starting the console application, it is necessary to have exactly two rows in the table – the header row and the data row.

9 Final start of the console application

After adding all the elements that should be in the intended print template and setting their parameters, it is possible to build different print templates. For example, below the image is an "rdlc" proforma invoice print template file.


Test - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Format Report Tools Test Window Help
Debug
conn
Solid
1 pt
Report5.rdlc [Design]
Server Explorer
Toolbox

Faktura
#ng_vydanafaktur
DAŇOVÝ DOKLAD

DATUM VYSTAVENÍ
31. 03. 2014
DATUM SPLATNOSTI
14. 04. 2014

Dodavatel
NetGenium s.r.o.
Anglická 414/28
120 00 Praha
IČ 27092381
DIČ CZ27092381

Odběratel
PLASTOKNO s.r.o.
Čisovice 72
252 04 Čisovice
IČ 26735377
DIČ CZ26735377

BANKOVNÍ ÚČET	VARIABILNÍ SYMBOL	ZPŮSOB PLATBY
2400213063/2010	120140213	Převodem

			DPH	Cena za MJ	Celkem bez DPH
			Header		
		#ng_polozkafaktury#.#ng_polozka#	Data		

Sazba DPH	Základ DPH	Výše DPH	Včetně DPH
DPH 0 %	500,00	0,00	500,00
DPH 16%	0,00	0,00	0,00

10 Incorporating a print template into an external function

Most of the files that take care of generating the print templates will have to be converted to external functions, which will then be uploaded to the given NET Genium.

The incorporation process is not just about adding new “rdlc” and “cs” files to the project (external functions). There are also various modifications to the “ngef.cs” class, which is always part of external functions, adding new libraries to the “References” folder, and editing the inserted class. Incorporating a print template can be done into a new or existing external function. The modifications to the “ngef.cs” class just differ in what external function it is.

It is also necessary to make some changes in the NET Genium quite often.

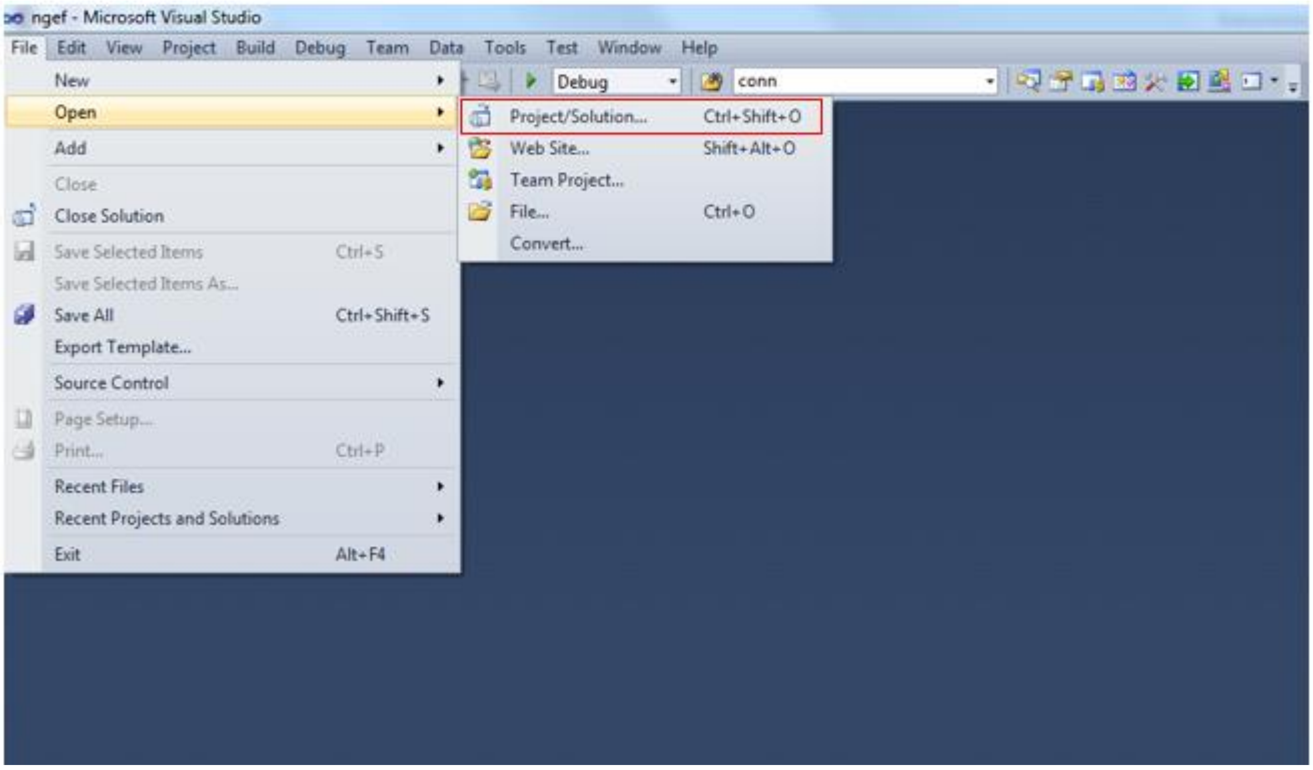
10.1 Procedure for a new external function (class “ngef.cs”)

10.1.1 Copy the project in the “References/ngef” directory to the customer folder

- An external function template is available in the “N:\NetGenium\Projects\NetGenium\ngef” directory. Copy this template to the directory of the customer for whom the new external function will be programmed.

10.1.2 Open the project

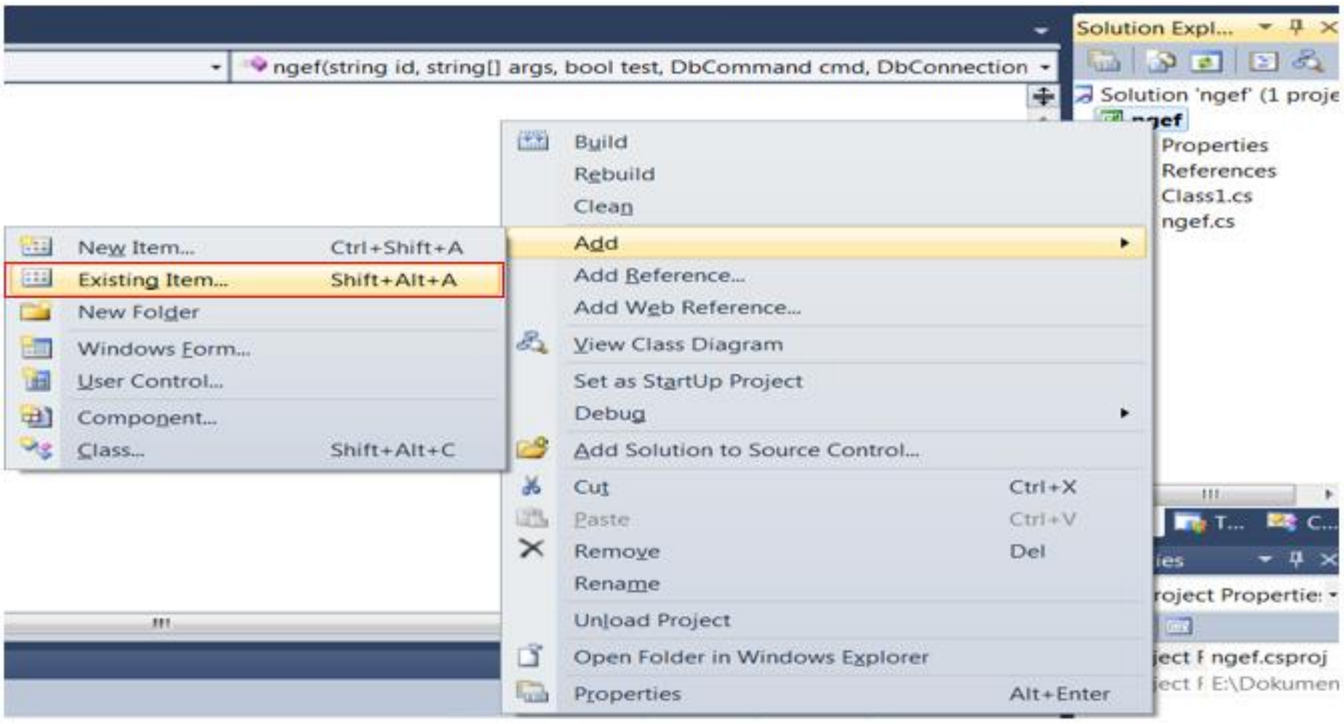
- Open the project by double-clicking on the “ngef.csproj” file in the copied “ngef” folder in the customer's directory.
- The second way to open a new external function is to open it using Visual Studio. For this method, select “File” – “Open” – “Project/Solution” after starting Visual Studio.


ngef - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools Test Window Help
New
Open
Add
Close
Close Solution
Save Selected Items Ctrl+S
Save Selected Items As...
Save All Ctrl+Shift+S
Export Template...
Source Control
Page Setup...
Print... Ctrl+P
Recent Files
Recent Projects and Solutions
Exit Alt+F4
Project/Solution... Ctrl+Shift+O
Web Site... Shift+Alt+O
Team Project...
File... Ctrl+O
Convert...
Debug
conn

- Then enter the path to the project. The project you are looking for will be the project that was copied to the customer folder in the previous step.

10.1.3Add the required class to the project

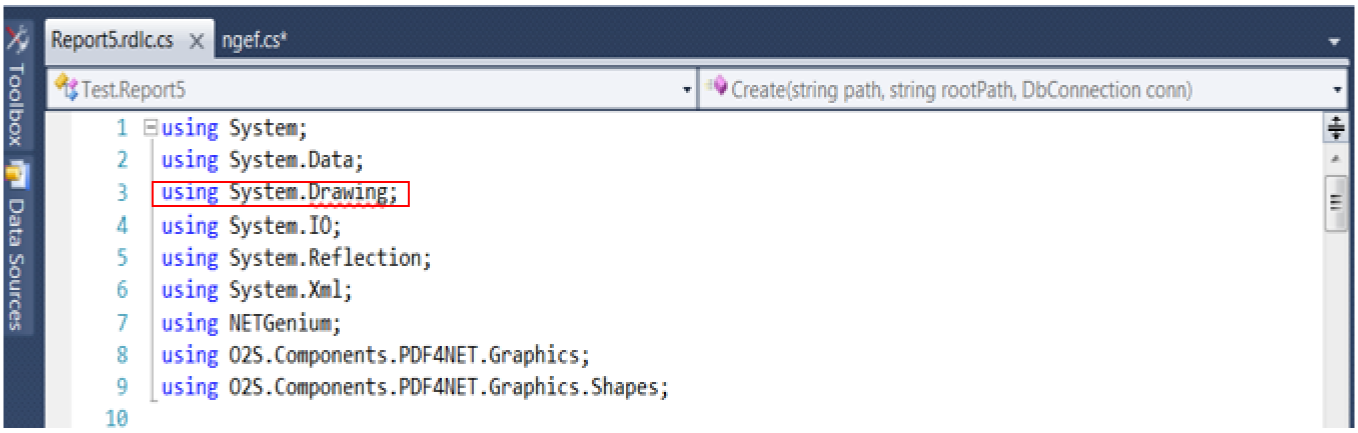
- To add a class created by the console application to an external function, right-click on the project name and select "Add" – "Existing Item".



- Then a window will open in which you enter the path to the required file. The added file will be a file with the extension "cs", which was created by launching our console application.

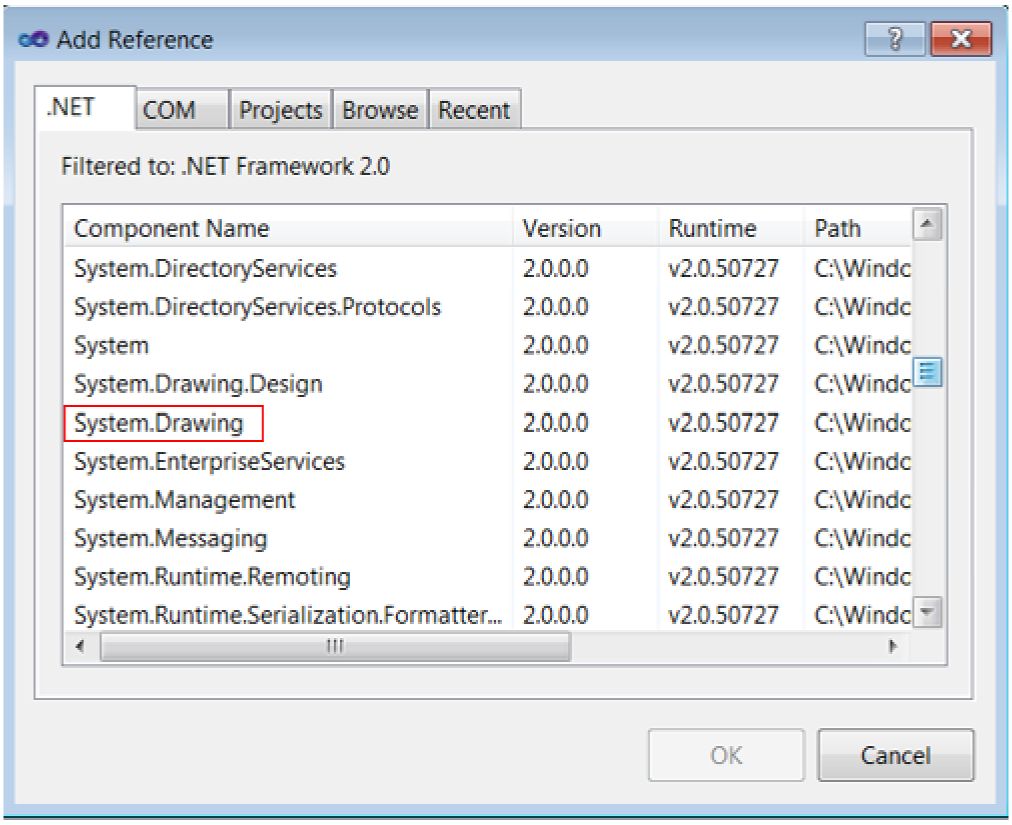
10.1.4Reference

- *In the case of creating a new external function, in addition to the libraries that have already been used in the console application – "NETGeniumConnection.dll" and "O2S.Components.PDF4NET.dll" – it is necessary to add the library "System.Drawing.dll". This library is listed after the "using" clause, but because it is not listed in the "References" folder, Visual Studio would report a syntax error when compiling the external function.*



```
using System;
using System.Data;
using System.Drawing;
using System.IO;
using System.Reflection;
using System.Xml;
using NETGenium;
using O2S.Components.PDF4NET.Graphics;
using O2S.Components.PDF4NET.Graphics.Shapes;
```

Uploading the "System.Drawing.dll" library to the project is done in the same way as described several times here. The only small change will be that when entering the path to the library, it will not click on the "Browse" tab but on ".NET". In this tab, you only need to find the one you want from the long list of libraries.

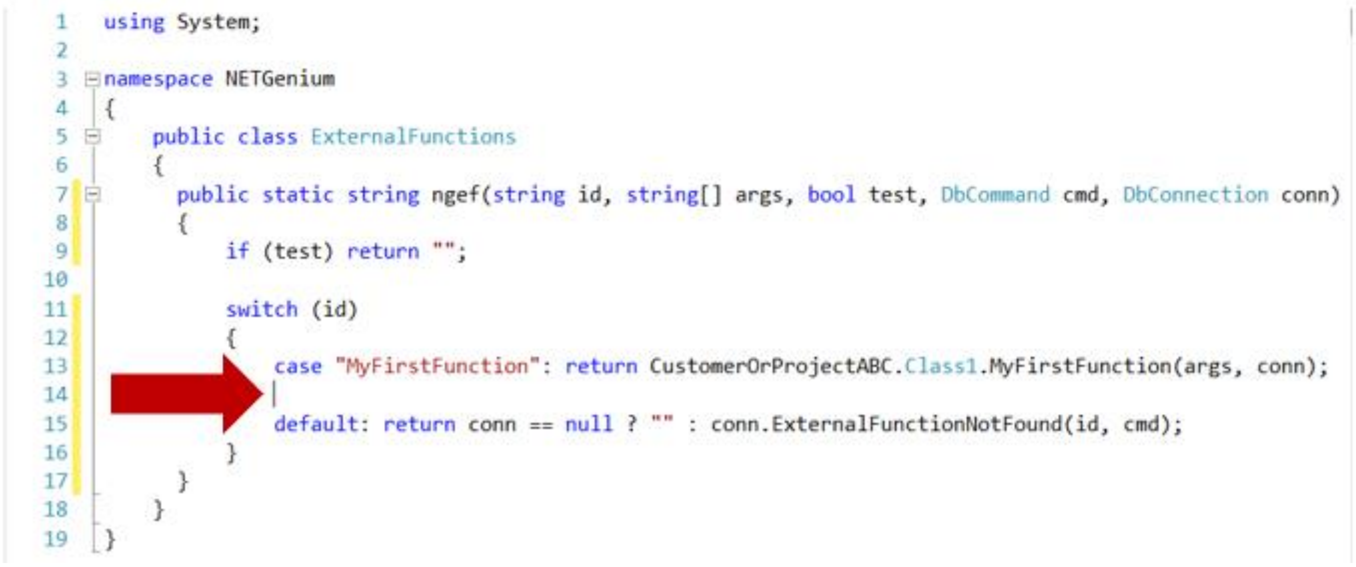


Filtered to: .NET Framework 2.0

Component Name	Version	Runtime	Path
System.DirectoryServices	2.0.0.0	v2.0.50727	C:\Windc
System.DirectoryServices.Protocols	2.0.0.0	v2.0.50727	C:\Windc
System	2.0.0.0	v2.0.50727	C:\Windc
System.Drawing.Design	2.0.0.0	v2.0.50727	C:\Windc
System.Drawing	2.0.0.0	v2.0.50727	C:\Windc
System.EnterpriseServices	2.0.0.0	v2.0.50727	C:\Windc
System.Management	2.0.0.0	v2.0.50727	C:\Windc
System.Messaging	2.0.0.0	v2.0.50727	C:\Windc
System.Runtime.Remoting	2.0.0.0	v2.0.50727	C:\Windc
System.Runtime.Serialization.Formatter...	2.0.0.0	v2.0.50727	C:\Windc

10.1.5Edit class “ngef.cs”

The “ngef.cs” class is mainly used to call methods that are offered by external functions. The body of the pure external function should now look like this.



```
using System;

namespace NETGenium
{
    public class ExternalFunctions
    {
        public static string ngef(string id, string[] args, bool test, DbCommand cmd, DbConnection conn)
        {
            if (test) return "";

            switch (id)
            {
                case "MyFirstFunction": return CustomerOrProjectABC.Class1.MyFirstFunction(args, conn);
                |
                default: return conn == null ? "" : conn.ExternalFunctionNotFound(id, cmd);
            }
        }
    }
}
```

In the place where the cursor and the red arrow are visible, insert the code that will be used to call the “Create ()” method of the inserted class, which will ensure the generation of the “pdf” document.

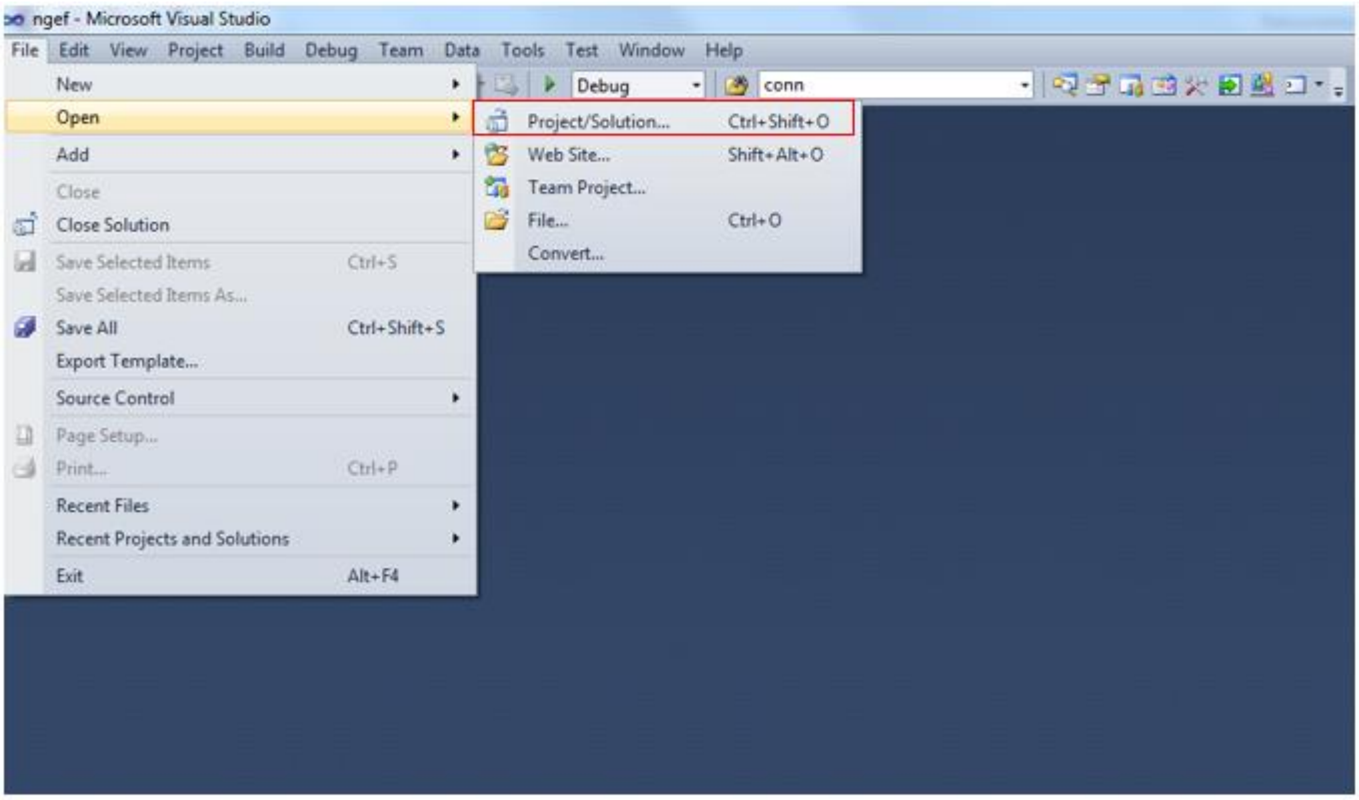
```
case "NETGenium.Print":
{
  if (conn.PrintingProcess.Template == "Invoice.pdf")
  {
    conn.PrintingProcess.FileName = Test.Report5.Create(conn.PrintingProcess.FilePath,
Config.RootPath, conn) + ".pdf";
  }
  return "";
}
```

The "Template" attribute of the "PrintingProcess" class identifies the template according to which the required document is generated, which can then be printed. In NET Genium, the template must always be uploaded. In practice, this means that you just need to save an empty "pdf" file, for example called "Invoice.pdf" in the "Templates" directory of the NET Genium. Further in the code, it is necessary that the string string "Invoice.pdf", which is given in the condition, exactly match the name of the file, which is uploaded in NET Genium in the directory "Templates". Otherwise, the condition would not be met and the correct template would not be generated.
If the condition is met, the method of the class that was created by the Report Designer may be called. The called method ensures that the required document is generated according to the code that is inside this method. The methods usually return the file name, which is stored in the "FileName" attribute of the "PrintingProcess" class.
As can be seen in the figure, each method that generates print output in "pdf" format has input parameters. The first parameter is the path where the newly generated document is to be saved, the second parameter is a string that specifies the path to the used NET Genium, the last parameter is the reference of the database from which the data is obtained.

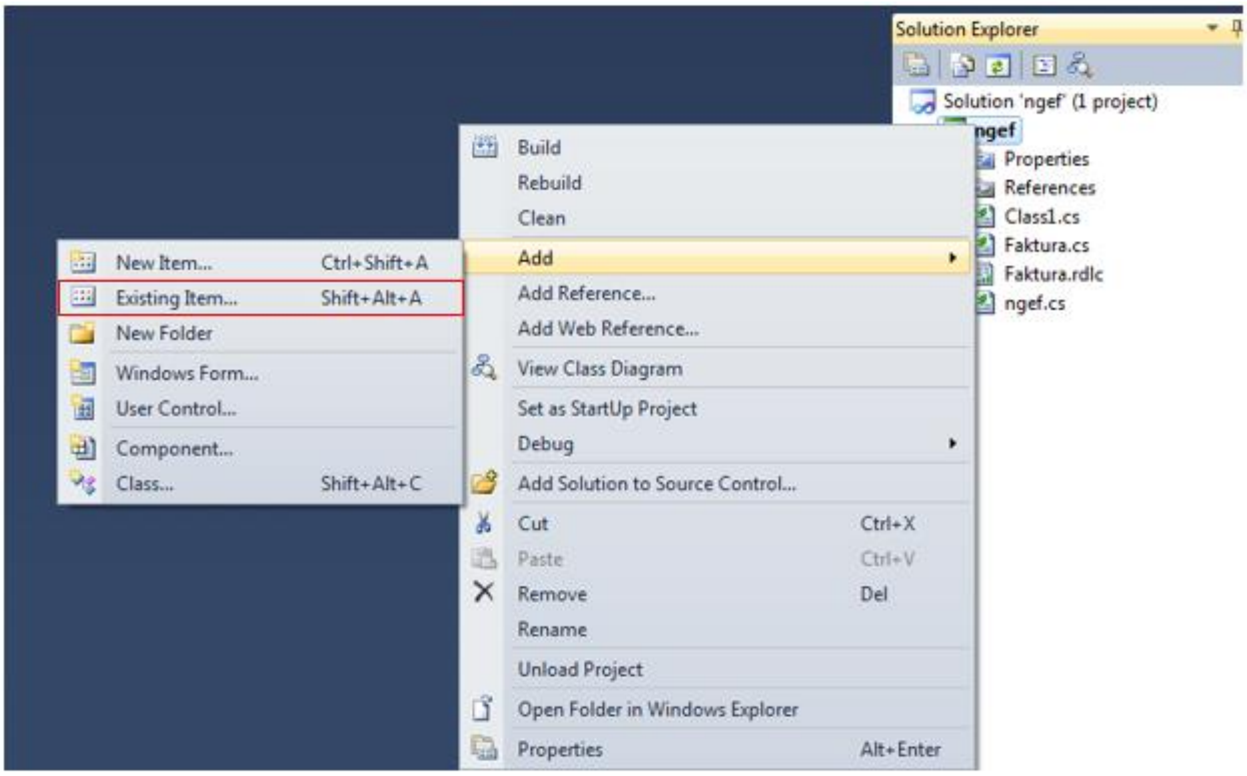
10.2 Procedure for an existing external function (class "ngef.cs")

10.2.1 Open the specified project where the external function is given

After starting Visual Studio, select "File" – "Open" – "Project/Solution…"



- Then enter the path to the project. Right-click on the project name and select “Add” – “Existing Item”



- Then a window will open in which you enter the path to the required file. The file to be added will be a file with the extension “cs”, which was created by launching our console application.

10.2.2Reference

Of course, all missing libraries that are used by the newly inserted class need to be uploaded to the project. The procedure is exactly the same as embedding libraries in a console application. In most cases, the missing library will be “O2S.Components.PDF4NET.dll”.

10.2.3 Edit the “ngef.cs” class in which the newly added class will be called

The project includes the “ngef.cs” class, which is used to call external function methods in NET Genium. A “case” condition is used in the “ngef.cs” class. A variant is designated for printing in NET Genium, which is marked as “NETGenium.Print”.

Add the following code to the “NETGenium.Print” identifier:

```
case "NETGenium.Print":
{
  if (conn.PrintingProcess.Template == "Invoice.pdf")
  {
    conn.PrintingProcess.FileName = Test.Report5.Create(conn.PrintingProcess.FilePath,
Config.RootPath, conn) + ".pdf";
  }
  return "";
}
```

The “Template” attribute of the “PrintingProcess” class identifies the template according to which the required document is generated, which can then be printed. In NET Genium, the template must always be uploaded. In practice, this means that you just need to save an empty “pdf” file, for example called “Invoice.pdf” in the “Templates” directory of the NET Genium. Further in the code, it is necessary that the string string “Invoice.pdf”, which is given in the condition, exactly match the name of the file, which is uploaded in NET Genium in the directory “Templates”. Otherwise, the condition would not be met and the correct template would not be generated.
If the condition is met, the method of the class that was created by the Report Designer may be called. The called method ensures that the required document is generated according to the code that is inside this method. The methods usually return the file name, which is stored in the “FileName” attribute of the “PrintingProcess” class.
As can be seen in the figure, each method that generates print output in “pdf” format has input parameters. The first parameter is the path where the newly generated document is to be saved, the second parameter is a string that specifies the path to the used NET Genium, the last parameter is the reference of the database from which the data is obtained.

10.3 Edit a newly inserted class

After incorporating the new class into the existing class “ngef.cs”, the code in the newly inserted class needs to be modified so that the class behaves exactly as expected – the same “pdf” document is generated as it was designed in Report Designer “rdlc” file.

10.3.1 Namespace change

First of all, change the namespace of the class, because it will definitely not match the namespace of the whole project.

10.3.2 Change the name of the embedded class

It is also a good practice to name classes according to what the class should represent. So, for example, if an invoice is to be generated, it's a good idea to call this class “Invoice. cs”.

10.3.3Change the name of the output file

- It is also a good idea to change the string that the embedded class method should return. Changing the string affects the name of the output file.

```
pdf.Save(path);
return "Faktura";
```

10.3.4Adjust image rendering

- *If images will appear in the template design, it is important to modify the code in the embedded class according to the following image.*

```
using (Stream stream = Assembly.GetExecutingAssembly().GetManifestResourceStream("ngef.Faktura.rdlc"))
using (Image image = pdf.GetImage(stream, "hlavicka"))
    if (image != null)
    {
        pdf.DrawImage(image, 93.3782, 32.3115, 44.7789, 25.1093);
    }
```

- *The first line and parameter of the “GetManifestResourceStream()” method is important. This parameter must be modified because this method will search the project for the file specified in the method parameter. The file you are looking for will be the “rdlc” file. If the method cannot find this file in the project, then when trying to generate a “pdf” document in NET Genium, an unhandled exception will occur and the generation will end. The first part of the parameter specifies the name of the library (assembly). In our example, we are editing a project (external function) called “ngef”. In the next section, the folders in which the file is stored within the project can be specified. The exact file name with the extension is always given at the end. In our case it is the file “Invoice.rdlc”. The project name, individual folders, and file name are separated by periods. It is necessary to add that the mentioned file “rdlc” must be added to the project using Visual Studio. The procedure for adding existing classes has already been mentioned in the text.*

10.3.5Change SQL queries

- *If a table with exactly two rows appears in the Report Designer, an SQL query is generated when the console application is started. However, this query must be modified in an external function, because Report Designer adds the default table “ng_form” to the query, and after clicking the print button, an exception would occur stating that the table “ng_form” does not exist in the database. For this reason, you need to go through all the SQL queries in the code and make sure that all the tables that are listed in the code really exist in the database.*

```
using (DataTable data = Data.Get("SELECT * FROM ng_form WHERE pid = " + conn["id"] + " ORDER BY id", conn))
```

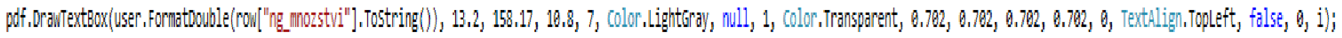
10.3.6Customize “TextBoxes” for field content

- *The content of inserted database fields or variables in the “TextBox” may in some cases exceed the defined dimensions (height and width) of the given “TextBox”. The procedure for setting the “TextBoxes” is different for the classically rendered “TextBox” and the “TextBox” rendered in the table.*

10.3.6.1 "TextBox" rendered in a table

The "TextBox" drawn in the table is identified by the arguments given in the "pdf.DrawTextBox()" method.

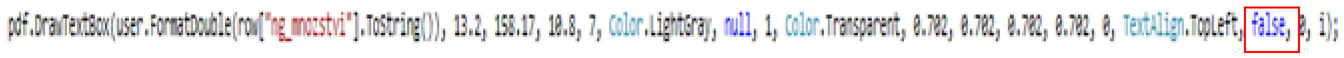
As can be seen in the figure below, the last parameter must be the "i" argument of type "int". By default, this argument is set to "-1", but in the case of a "TextBox" plotted in a table, the value is "0" or "1" (depending on the item cycle). Furthermore, the "pdf.DrawTextBox()" method contains the "Clip" argument, which is always set to "false" in the case of a "TextBox" drawn in the table after generation. In the case of a "TextBox" drawn in a table, it is advisable not to change the value of the "Clip" argument and leave it set to "false".



```
pdf.DrawTextBox(user.FormatDouble(row["ng_mnozstvi"].ToString()), 13.2, 158.17, 10.8, 7, Color.LightGray, null, 1, Color.Transparent, 0.702, 0.702, 0.702, 0.702, 0, TextAlign.TopLeft, false, 0, i);
```

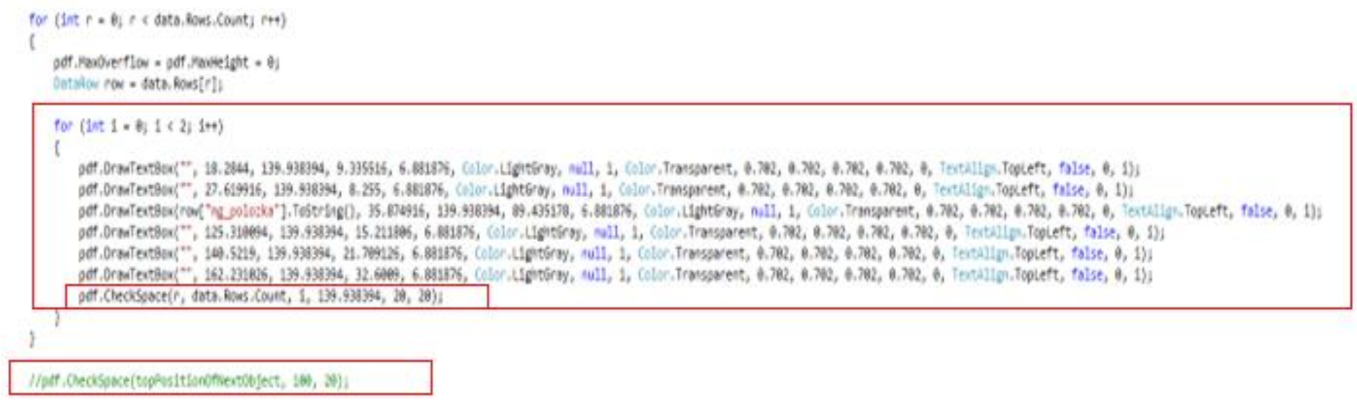
10.3.6.2 Clip attribute

The "Clip" attribute is of type "boolean". If the "Clip" argument is set to "false", then the defined height of the "TextBox" from Visual Studio is allowed to overflow. When the content of a given "TextBox" is longer than the width of the defined "TextBox", the content expands to several lines and the height of the defined "TextBox" adjusts to the content of the "TextBox".



```
pdf.DrawTextBox(user.FormatDouble(row["ng_mnozstvi"].ToString()), 13.2, 158.17, 10.8, 7, Color.LightGray, null, 1, Color.Transparent, 0.702, 0.702, 0.702, 0.702, 0, TextAlign.TopLeft, false, 0, i);
```

10.3.6.3 Pdf.CheckSpace () method



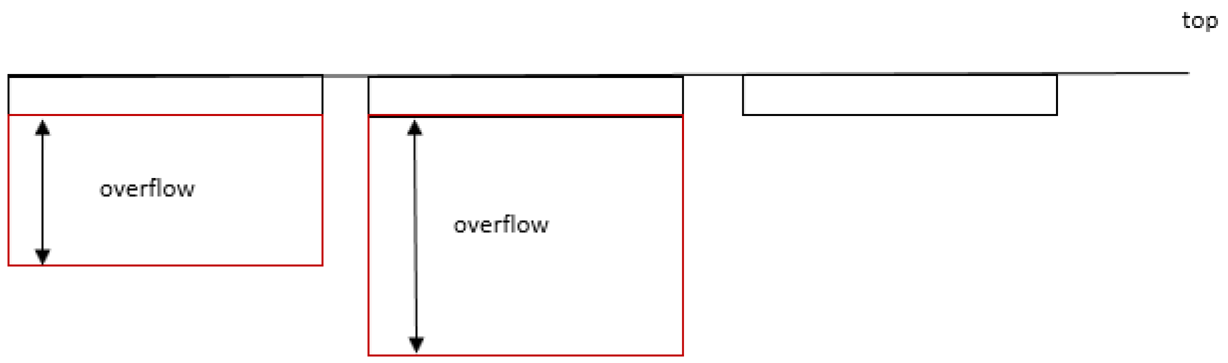
```
for (int r = 0; r < data.Rows.Count; r++)
{
    pdf.MaxOverflow = pdf.MaxHeight = 0;
    DataRow row = data.Rows[r];

    for (int i = 0; i < 2; i++)
    {
        pdf.DrawTextBox("", 18.2844, 139.938394, 9.335516, 6.881876, Color.LightGray, null, 1, Color.Transparent, 0.702, 0.702, 0.702, 0.702, 0, TextAlign.TopLeft, false, 0, i);
        pdf.DrawTextBox("", 27.619916, 139.938394, 8.255, 6.881876, Color.LightGray, null, 1, Color.Transparent, 0.702, 0.702, 0.702, 0.702, 0, TextAlign.TopLeft, false, 0, i);
        pdf.DrawTextBox(row["ng_polozka"].ToString(), 35.874916, 139.938394, 89.435178, 6.881876, Color.LightGray, null, 1, Color.Transparent, 0.702, 0.702, 0.702, 0.702, 0, TextAlign.TopLeft, false, 0, i);
        pdf.DrawTextBox("", 125.310094, 139.938394, 15.211806, 6.881876, Color.LightGray, null, 1, Color.Transparent, 0.702, 0.702, 0.702, 0.702, 0, TextAlign.TopLeft, false, 0, i);
        pdf.DrawTextBox("", 140.5219, 139.938394, 21.709126, 6.881876, Color.LightGray, null, 1, Color.Transparent, 0.702, 0.702, 0.702, 0.702, 0, TextAlign.TopLeft, false, 0, i);
        pdf.DrawTextBox("", 162.231026, 139.938394, 32.6009, 6.881876, Color.LightGray, null, 1, Color.Transparent, 0.702, 0.702, 0.702, 0.702, 0, TextAlign.TopLeft, false, 0, i);
        pdf.CheckSpace(r, data.Rows.Count, i, 139.938394, 20, 20);
    }
}

//pdf.CheckSpace(topPositionOfNextObject, 100, 20);
```

As you can see in the image above, two "pdf.CheckSpace()" methods are defined in the code that renders all the items stored for a given invoice. These methods determine if the next plotted element will not be its y-coordinate outside the drawing area of the document. If so, the methods ensure that the next rendered element is moved to the next page. The penultimate parameter of the method says how much space is needed for the footer of the page. The last parameter specifies how much space should be left for the page header when the next page is jumped.

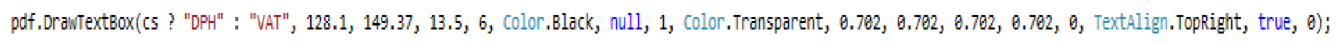
In addition to the functionality described above, the first overloaded method also ensures that the following elements are moved along the y-axis according to the maximum overflow. As can be seen in the figure above, one invoice item is drawn in two cycles. The "pdf.CheckSpace()" method is called in each of these cycles. The figure below shows an example of the layout of "TextBoxes" of one item. In the first cycle, the "TextBox" with the largest overflow value of the defined “TextBox” is identified and this value is stored in the “MaxOverflow” variable. The individual fields of the item are not rendered in the first cycle. In the second cycle, all “TextBoxes” of the item are set to the “height” attribute to a value that is equal to the value of the “MaxOverflow” attribute obtained in the first cycle. Resetting the “height” attribute ensures that all fields in one item have the same height, and the height of the previous item is taken into account when rendering the next item, so that the individual items do not overlap.



The second method is commented by default because it requires the y-coordinate of the next object below the table as the first parameter. If necessary, the value can be read from the code. The second case of the method is used especially when the cycle of browsing individual items ends and it is necessary to verify whether there is still room for the next text of the document on the same page.

10.3.6.4 Classically rendered “TextBox”

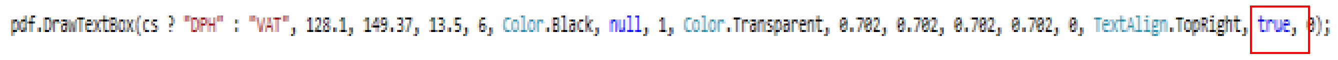
The “pdf.DrawTextBox()” method of the classically rendered “TextBox” contains fewer arguments than the “TextBox” rendered in the table. The arguments lack the “int i” argument, which is always set to “-1” in the case of a classically rendered “TextBox”. For a classically rendered “TextBox”, the “Clip” argument is set to “true” after the code is generated. This argument can be changed manually as needed.



```
pdf.DrawTextBox(cs ? "DPH" : "VAT", 128.1, 149.37, 13.5, 6, Color.Black, null, 1, Color.Transparent, 0.702, 0.702, 0.702, 0.702, 0, TextAlign.TopRight, true, 0);
```

10.3.6.5 Clip attribute

The “Clip” attribute is of type “Boolean”. If the “Clip” argument is set to “true”, the height of the “TextBox” will be rendered in exactly the same way as it was designed in Report Designer in Visual Studio without the possibility of overflow. This means that a “TextBox” with content that exceeds the defined width of the “TextBox” will be drawn on one line, and therefore the entire content of the field may not be rendered in the “pdf” document.



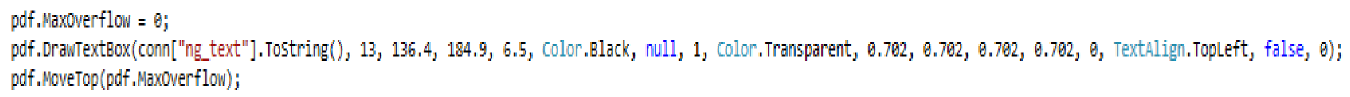
```
pdf.DrawTextBox(cs ? "DPH" : "VAT", 128.1, 149.37, 13.5, 6, Color.Black, null, 1, Color.Transparent, 0.702, 0.702, 0.702, 0.702, 0, TextAlign.TopRight, true, 0);
```

10.3.6.6 MaxOverflow attribute

The figure below shows how to ensure that when the “Clip” attribute is set to “false” in the classically rendered “TextBox”, the following elements are shifted by the y-coordinate. It is necessary to reset the “MaxOverflow” attribute, because each call to the “pdf.DrawTextBox()” method sets the value in this attribute and it is necessary to have it reset before setting the offset of the following elements.

10.3.6.6.1 Pdf.MoveTop (double overflow) method

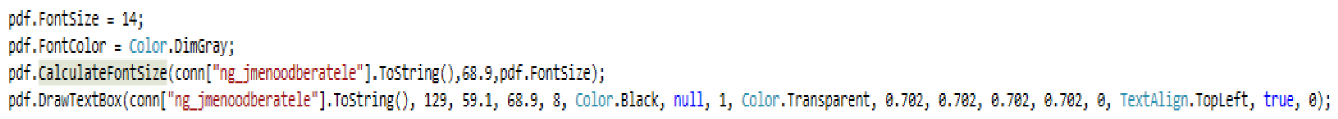
The value of the “pdf.MaxOverflow” attribute, which is set in the “pdf.DrawTextBox()” method, is inserted as an argument to this method. Calling the method ensures that the “pdf.DrawTextBox()” method of the next plotted element shifts along the y-new axis according to the value “pdf.MaxOverflow”. If a case arises in which several classically rendered “TextBoxes” are compared side by side with the value of the “Clip” attribute set to “false”, it is sufficient to insert the “pdf.DrawTextBox()” method of the given “TextBoxes” among the settings of the “pdf.MaxOverflow ” attribute and the “pdf.MoveTop(pdf.MaxOverflow)”method.



```
pdf.MaxOverflow = 0;
pdf.DrawTextBox(conn["ng_text"].ToString(), 13, 136.4, 184.9, 6.5, Color.Black, null, 1, Color.Transparent, 0.702, 0.702, 0.702, 0.702, 0, TextAlign.TopLeft, false, 0);
pdf.MoveTop(pdf.MaxOverflow);
```

10.3.6.6.2 Pdf.CalculateFontSize () method

If the value of the “Clip” attribute is set to “true” for the classically rendered “TextBox” (the lines will not be wrapped) and the content of the inserted field or variable is so long that it does not fit within the defined width of the “TextBox”, available method “pdf.CalculateFontSize(object value, double maxWidth, int size)”, which ensures the reduction of the defined font size until the content fits within the width of the defined “TextBox”.



```
pdf.FontSize = 14;
pdf.FontColor = Color.DimGray;
pdf.CalculateFontSize(conn["ng_jmenoodberatele"].ToString(),68.9,pdf.FontSize);
pdf.DrawTextBox(conn["ng_jmenoodberatele"].ToString(), 129, 59.1, 68.9, 8, Color.Black, null, 1, Color.Transparent, 0.702, 0.702, 0.702, 0.702, 0, TextAlign.TopLeft, true, 0);
```

10.4 Creating forms in NET Genium

As already written in the introduction, the prerequisite for generating a “pdf” document is to have NET Genium installed on the local computer. The form creation procedure is designed for pure NET Genium. If the already used NET Genium is available, it depends on what your task is to create, but at least you will also use the point about creating a button for generating a print template. The Administrator's Guide can be used for the detailed procedure of creating forms in NET Genium.

Simplified procedure

1. Open the installed NET Genium on the local computer.
2. Create a new “Finance” application group.
3. Create a new “Issued Invoices” application.
4. Create new edit forms “Issued invoice” and “Issued invoice item”.
5. Check uploaded templates in NET Genium.
6. Create a new “Issued Invoices” view page.

10.4.1Open NET Genium

- First of all, of course, the most important thing is to run the installed NET Genium on the local computer. Enter "localhost/NET Genium virtual directory" as the browser URL. The virtual directory was already created when you installed NET Genium.

10.4.2Create application group "Finance"

- Because open NET Genium is clean, you need to create an application group first. The application group is used in NET Genium to merge applications that cover the same area. These areas can be, for example, finance, marketing, inventory, etc. An application group "Finance" is created for the purposes of this manual, but it is possible to create an application group with any name.

10.4.3Create a new "Issued Invoices" application

- It is also possible to create any number of applications in the created application group. Applications should be grouped into appropriate application groups. In the described example of this manual, the application "Issued invoices" was created in the application group "Finance". As with the application group, the application can have any name.

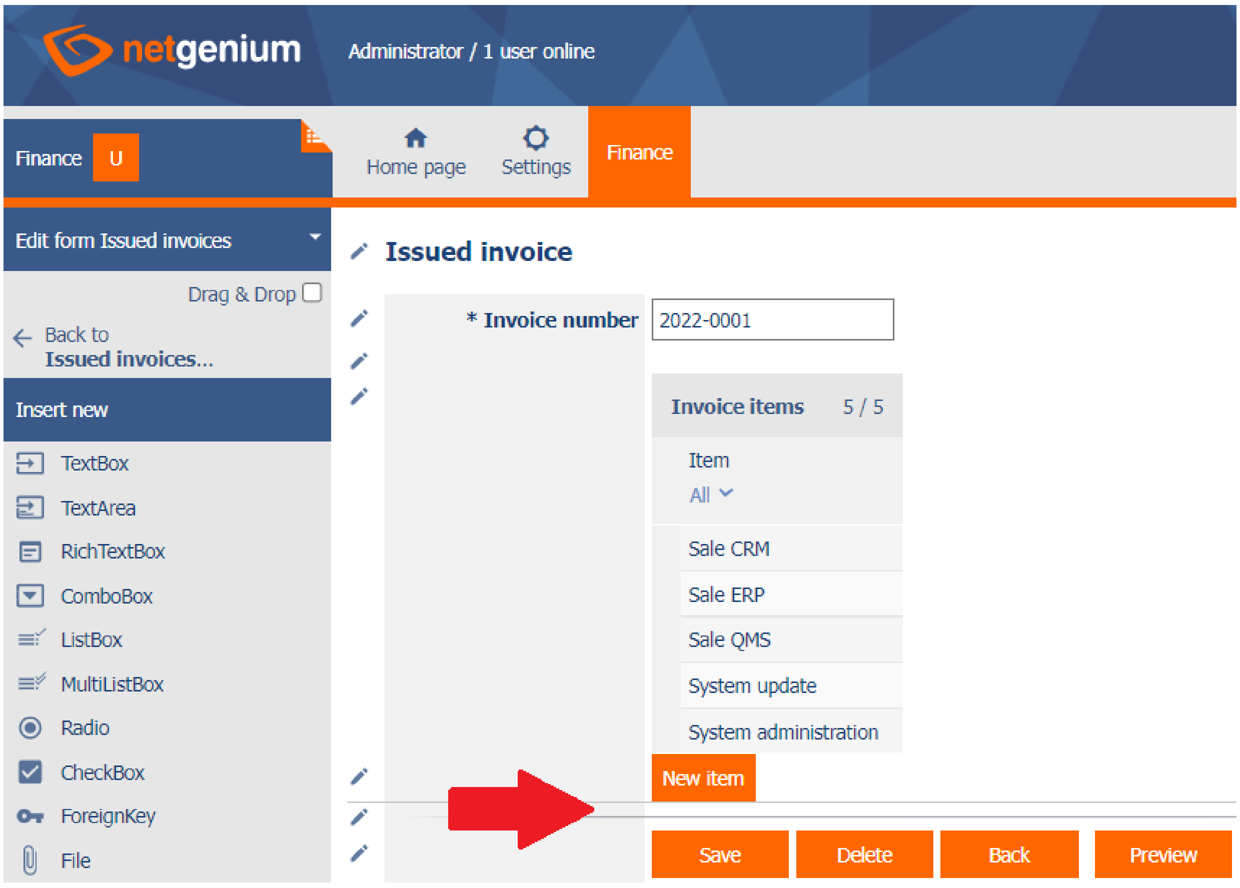
10.4.4Create new edit forms "Issued invoice" and "Issued invoice item"

- *For a short demonstration of loading the NET Genium edit form fields into the prepared template, two edit forms are created – "Issued invoice" and "Issued invoice item". The text field "Invoice number" is created in the edit form "Issued invoice", which is then loaded into the template under the database identifier "#ng_issuedinvoice#.#ng_invoicenumber#". One text field named "Item" is also created in the "Issued invoice item" edit form. This text box is then loaded in the template under the identifier "#ng_issuedinvoiceitem#.#ng_item#" in the table specified for each invoice item.*
  Furthermore, the edit forms are interconnected by means of a link via the identifiers "ng_issuedinvoiceitem.pid = ng_issuedinvoice.id".
  For the purposes of this guide, edit forms are very simple. In practice, the individual forms are much more complex because they contain more controls and links to other tables in the database.

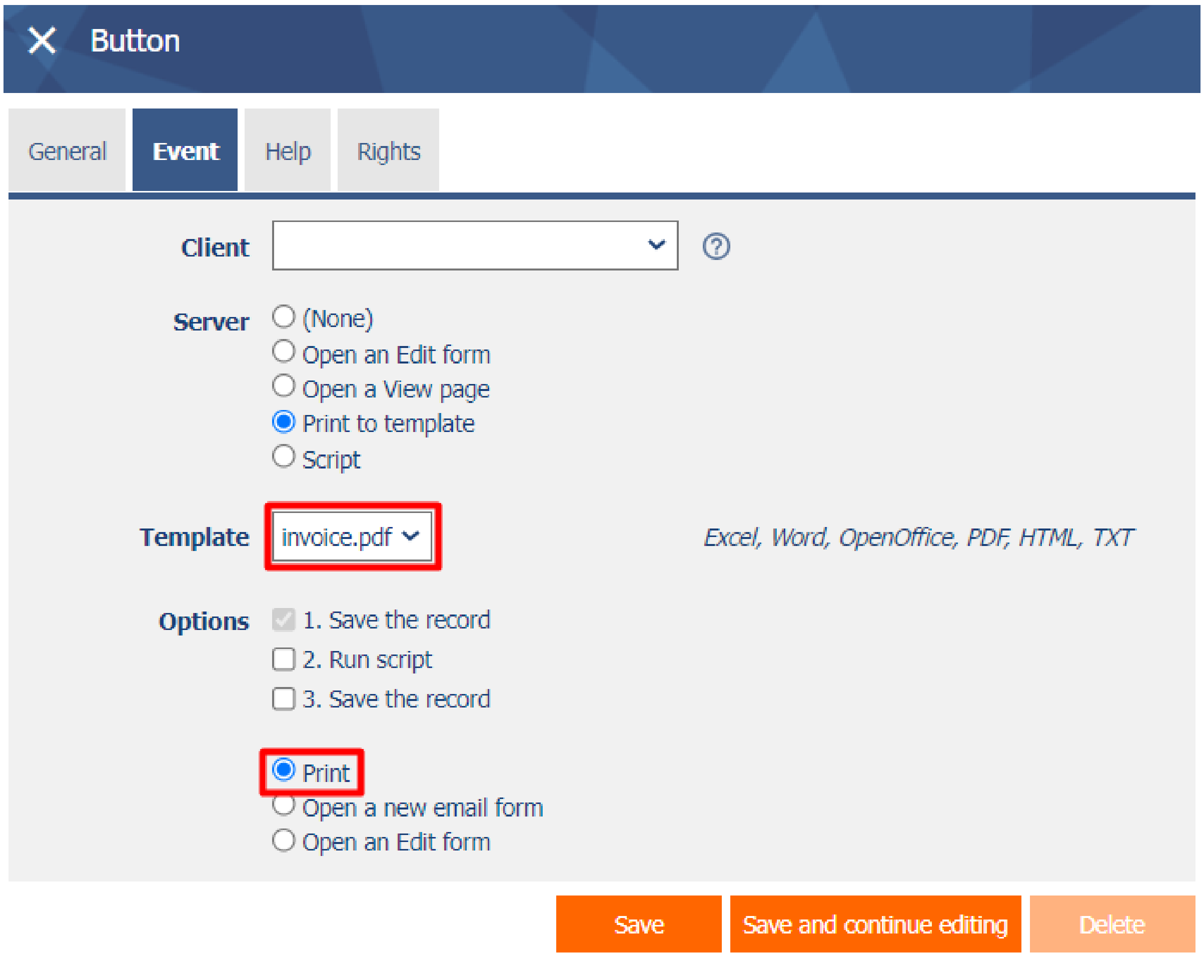
Inserting "Button"

- *When creating the "Issued invoice" edit form, it is necessary to point out one important thing related to the generation of templates. The control that is used to create "pdf" documents is the "Button" control.*

- Insert the button in the “Issued invoice” edit form in the position shown in the following figure.



- After inserting the mentioned control, the “Button” setting opens. Since this is a button that should perform an action, go to the “Events” tab after filling in the required “Name” field. If the user wants to generate a “pdf” document by pressing the button, it is necessary to select the “Print to template” option in the “Server” menu and have the “Print” option checked in the last part of the window, as shown in the figure below.


Button

General | **Event** | Help | Rights

Client

Server: ( ) (None) ( ) Open an Edit form ( ) Open a View page [✓ Print to template] ( ) Script

Template: invoice.pdf — Excel, Word, OpenOffice, PDF, HTML, TXT

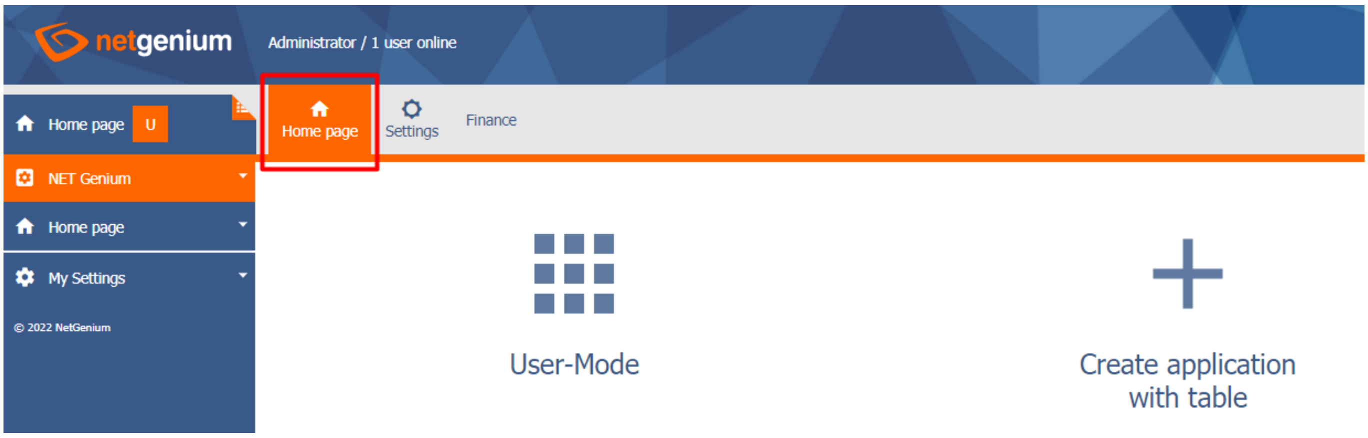
Options: [✓ 1. Save the record] ☐ 2. Run script ☐ 3. Save the record

[✓ Print] ( ) Open a new email form ( ) Open an Edit form

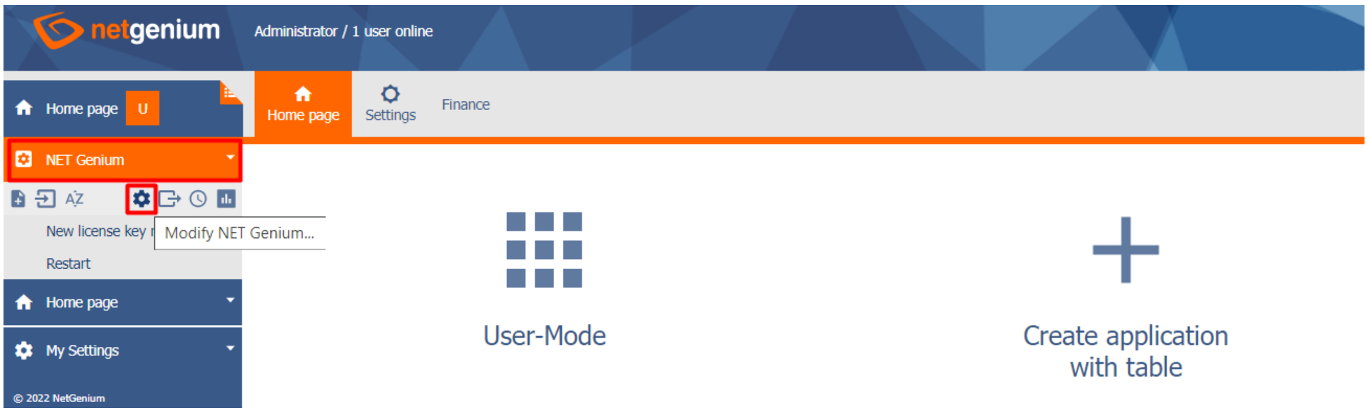
- Next, for the element named “Template”, specify according to which template a new document should be created. The menu will contain all the files that are stored in the “Templates” folder of the NET Genium.

If the required template is not listed in the menu, you must add the file to the specified folder. There are two ways to do this. One of them is to save a “pdf” file, for example called “Invoice.pdf”, directly to the “Templates” folder of the given NET Genium. The second way is to upload the file “Invoice.pdf” via the NET Genium system. To upload a template to NET Genium, it is necessary to proceed as follows:

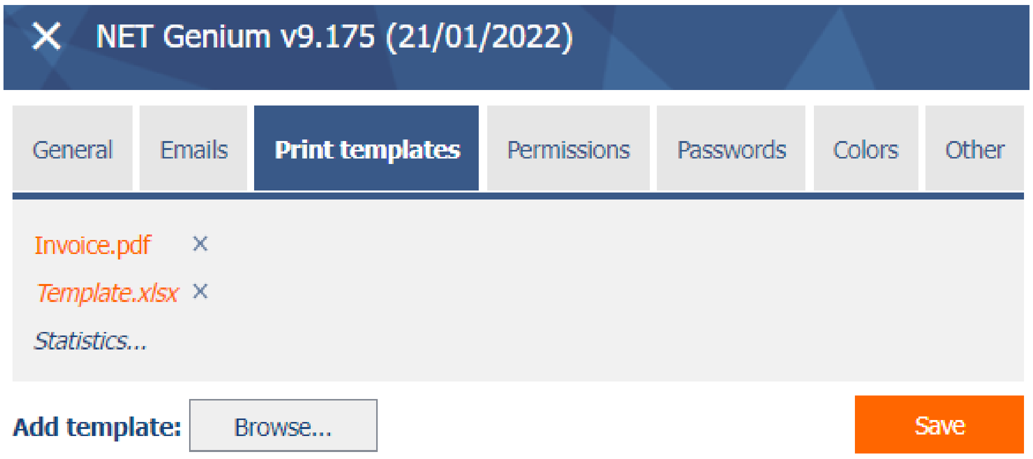
- Click on the “Home Page” tab.



- Click on the NET Genium main menu and select the “Modify NET Genium” option (gear wheel).



- Go to the “Printing Templates” tab.
- A window will open in which all saved templates in NET Genium are listed on the “Print templates” tab. You can upload a new print template by pressing the “Browse…” button. The print template name should match the name used in the code in the “ngef.cs” class of the external function.



10.4.5 Create a new “Issued Invoices” view page

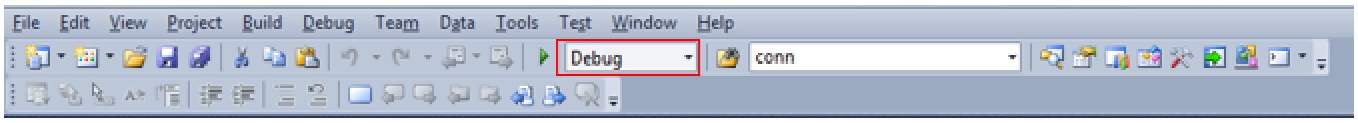
For the purposes of this manual, a view page “Issued invoices” is created, which contains a view table with records of all issued invoices. Of course, the user can create any number of view tables that can display various records from the database.

10.5 Uploading an external function to a local NET Genium

10.5.1 Compile the external function

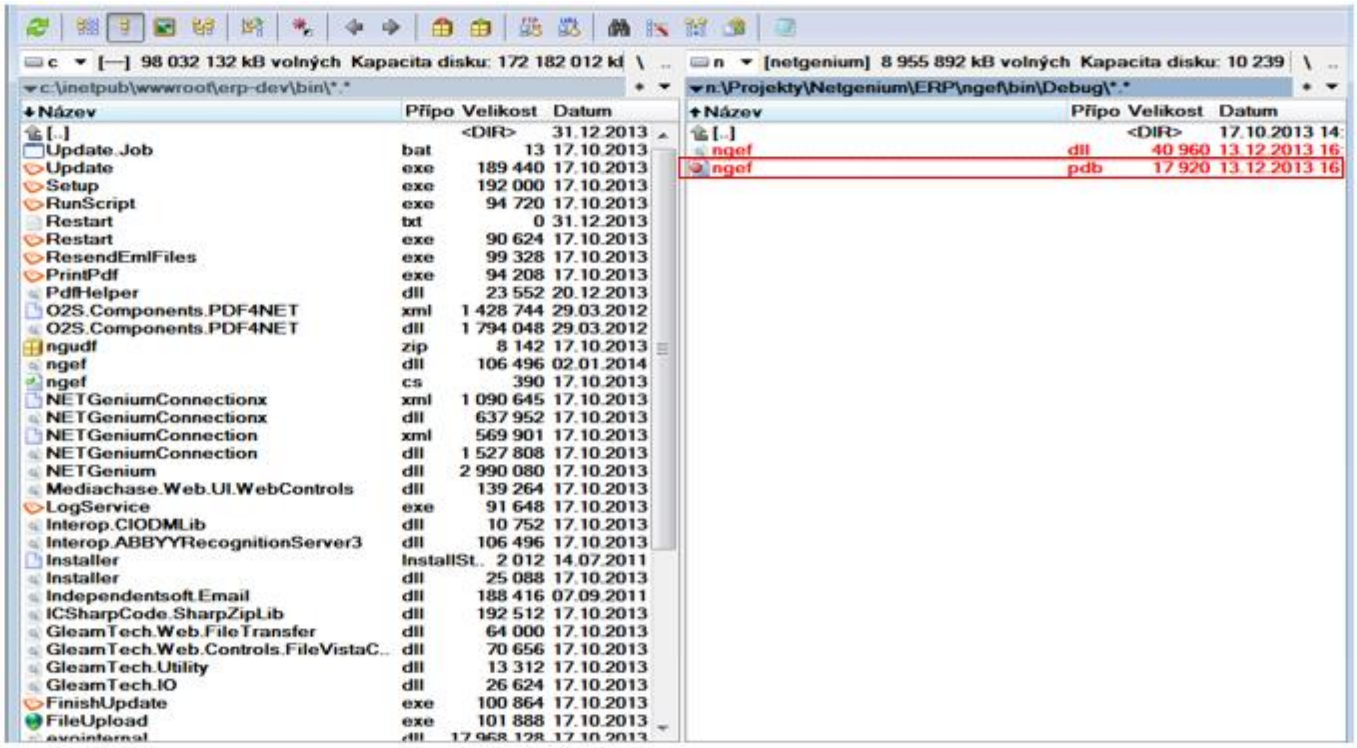
If there is nothing to edit in the classes, you can compile the external function using the keyboard shortcut “CTRL + SHIFT + B”. Two things will be compiled. First, it checks for any syntax errors in the code.

There are two “Debug” and “Release” modes in Visual Studio. The “Debug” mode is intended for testing, because after compiling the external function, it creates another file with the “pdb” extension in addition to the file with the “dll” extension. When an error occurs while an external function is running, the “pdb” file specifies exactly the line of the external function where the error occurred. You can switch between modes in the menu just below the main menu of Visual Studio. Each of these modes has its own folder in which the compiled external functions are stored.


File Edit View Project Build Debug Team Data Tools Test Window Help
Debug
conn

10.5.2Upload a new version of the external function to NET Genium

After opening the Total Commander program under administrator rights, the “dll” file created by compiling the external function is uploaded to the “bin” directory of the given NET Genium. The picture below shows that the external function was compiled in “Debug” mode. For this reason, a file with the “pdb” extension was generated, and therefore it is possible to upload a “pdb” file to NET Genium together with a file with the “dll” extension.



```
c  [—]  98 032 132 kB volných  Kapacita disku: 172 182 012 kl
c:\inetpub\wwwroot\erp-dev\bin\*.*
Název                            Přípo     Velikost  Datum
[..]                                          <DIR>  31.12.2013
Update.Job                       bat             13  17.10.2013
Update                           exe        189 440  17.10.2013
Setup                            exe        192 000  17.10.2013
RunScript                        exe         94 720  17.10.2013
Restart                          txt              0  31.12.2013
Restart                          exe         90 624  17.10.2013
ResendEmlFiles                   exe         99 328  17.10.2013
PrintPdf                         exe         94 208  17.10.2013
PdfHelper                        dll         23 552  20.12.2013
O2S.Components.PDF4NET           xml      1 428 744  29.03.2012
O2S.Components.PDF4NET           dll      1 794 048  29.03.2012
ngudf                            zip          8 142  17.10.2013
ngef                             dll        106 496  02.01.2014
ngef                             cs             390  17.10.2013
NETGeniumConnectionx             xml      1 090 645  17.10.2013
NETGeniumConnectionx             dll        637 952  17.10.2013
NETGeniumConnection              xml        569 901  17.10.2013
NETGeniumConnection              dll      1 527 808  17.10.2013
NETGenium                        dll      2 990 080  17.10.2013
Mediachase.Web.UI.WebControls    dll        139 264  17.10.2013
LogService                       exe         91 648  17.10.2013
Interop.CIODMLib                 dll         10 752  17.10.2013
Interop.ABBYYRecognitionServer3  dll        106 496  17.10.2013
Installer                        InstallSt.. 2 012  14.07.2011
Installer                        dll         25 088  17.10.2013
Independentsoft.Email            dll        188 416  07.09.2011
ICSharpCode.SharpZipLib          dll        192 512  17.10.2013
GleamTech.Web.FileTransfer       dll         64 000  17.10.2013
GleamTech.Web.Controls.FileVistaC.. dll      70 656  17.10.2013
GleamTech.Utility                dll         13 312  17.10.2013
GleamTech.IO                     dll         26 624  17.10.2013
FinishUpdate                     exe        100 864  17.10.2013
FileUpload                       exe        101 888  17.10.2013
```

```
n  [netgenium]  8 955 892 kB volných  Kapacita disku: 10 239
n:\Projekty\Netgenium\ERP\ngef\bin\Debug\*.*
Název                            Přípo     Velikost  Datum
[..]                                          <DIR>  17.10.2013 14:
ngef                             dll         40 960  13.12.2013 16:
ngef                             pdb         17 920  13.12.2013 16
```

10.5.3Update NET Genium

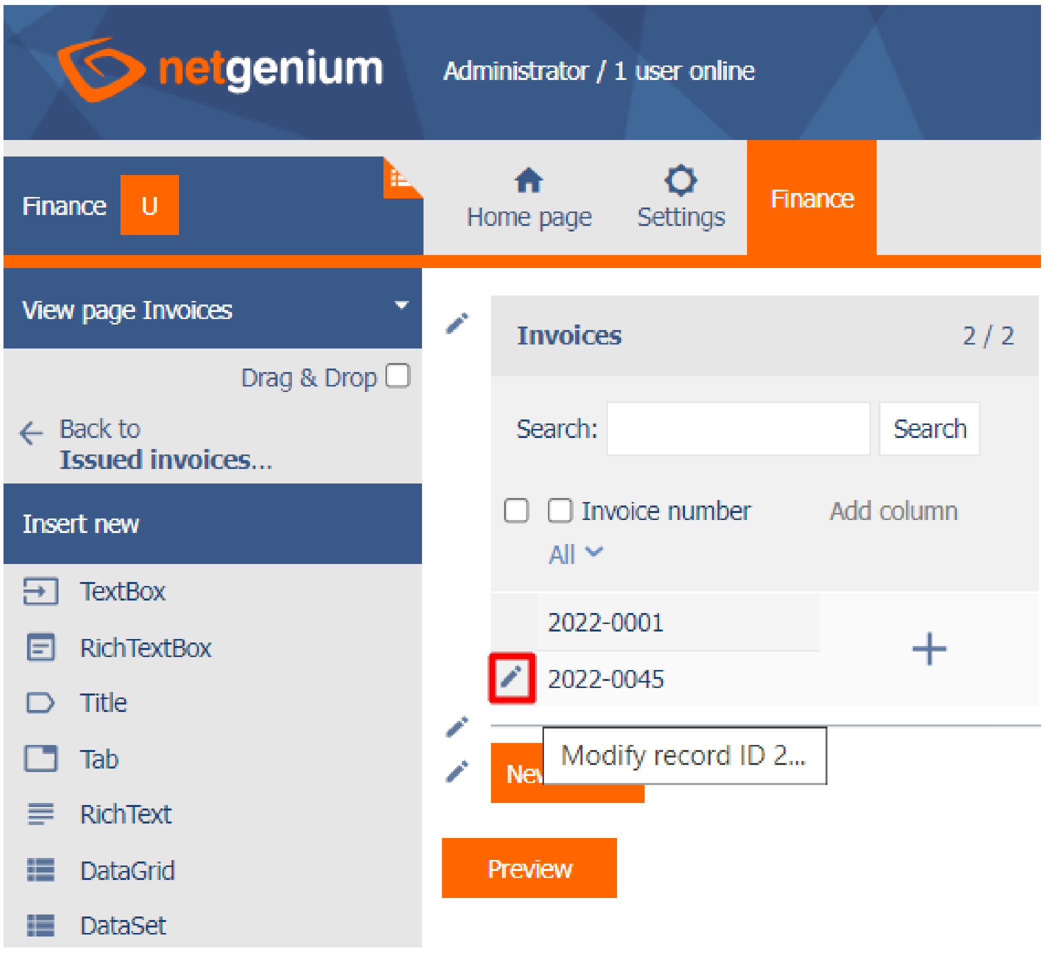
It is now possible to test the new functionality of the external function. In the open NET Genium, press the “F5” key to update the NET Genium.

10.5.4Enter login details

After the update, the system will require you to re-enter your credentials.

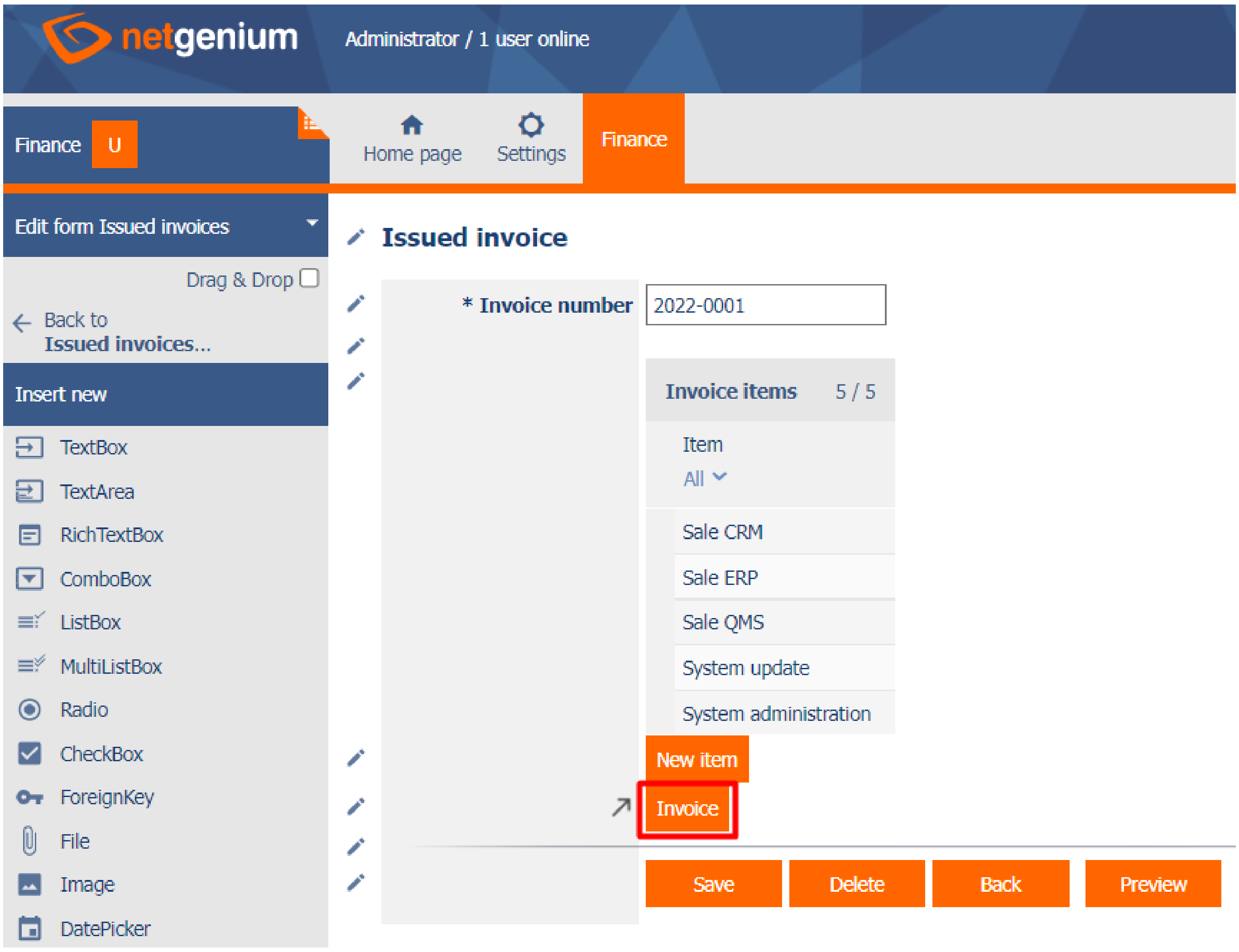
10.5.5Open the “Issued invoices” view page

- To print the document “Invoice.pdf”, open the edit form of a specific issued invoice. Already saved records are opened using view pages. Specifically for the case of our manual, it is necessary to open the view pages “Issued invoices”. Each record has a pencil icon that you can use to open the edit form for that record.



10.5.6Open the selected record of the issued invoice and press the “Invoice” button

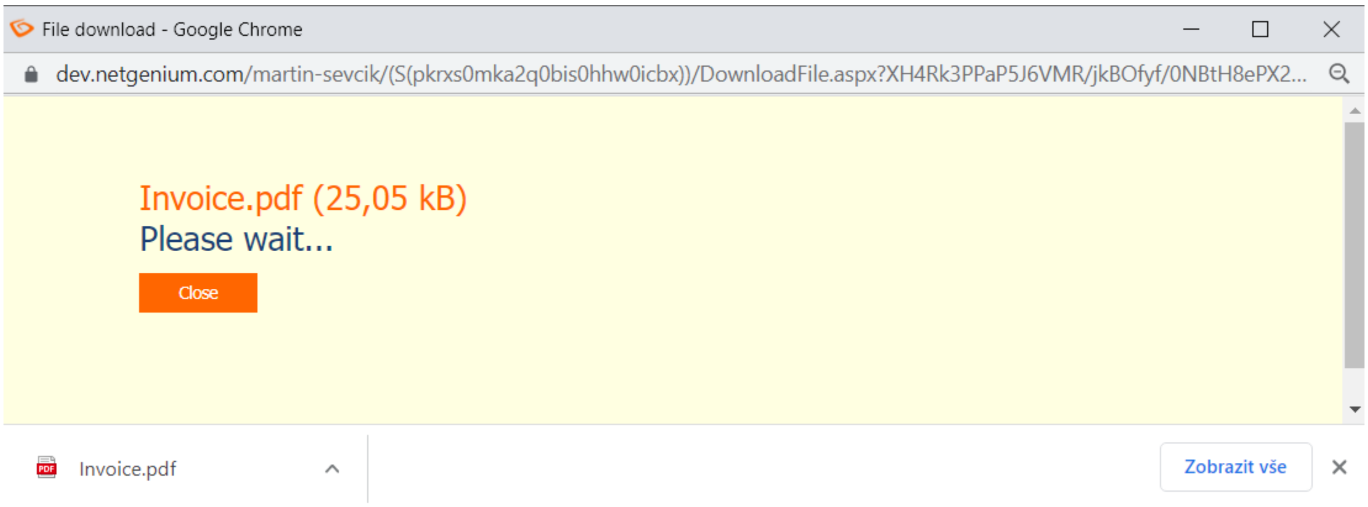
- Pressing the pencil icon next to the selected record in the “Issued invoices” view table opens the edit form of the selected record. To generate a print template “Invoice.pdf”, press the “Invoice” button, which will ensure the generation of this “pdf” document.



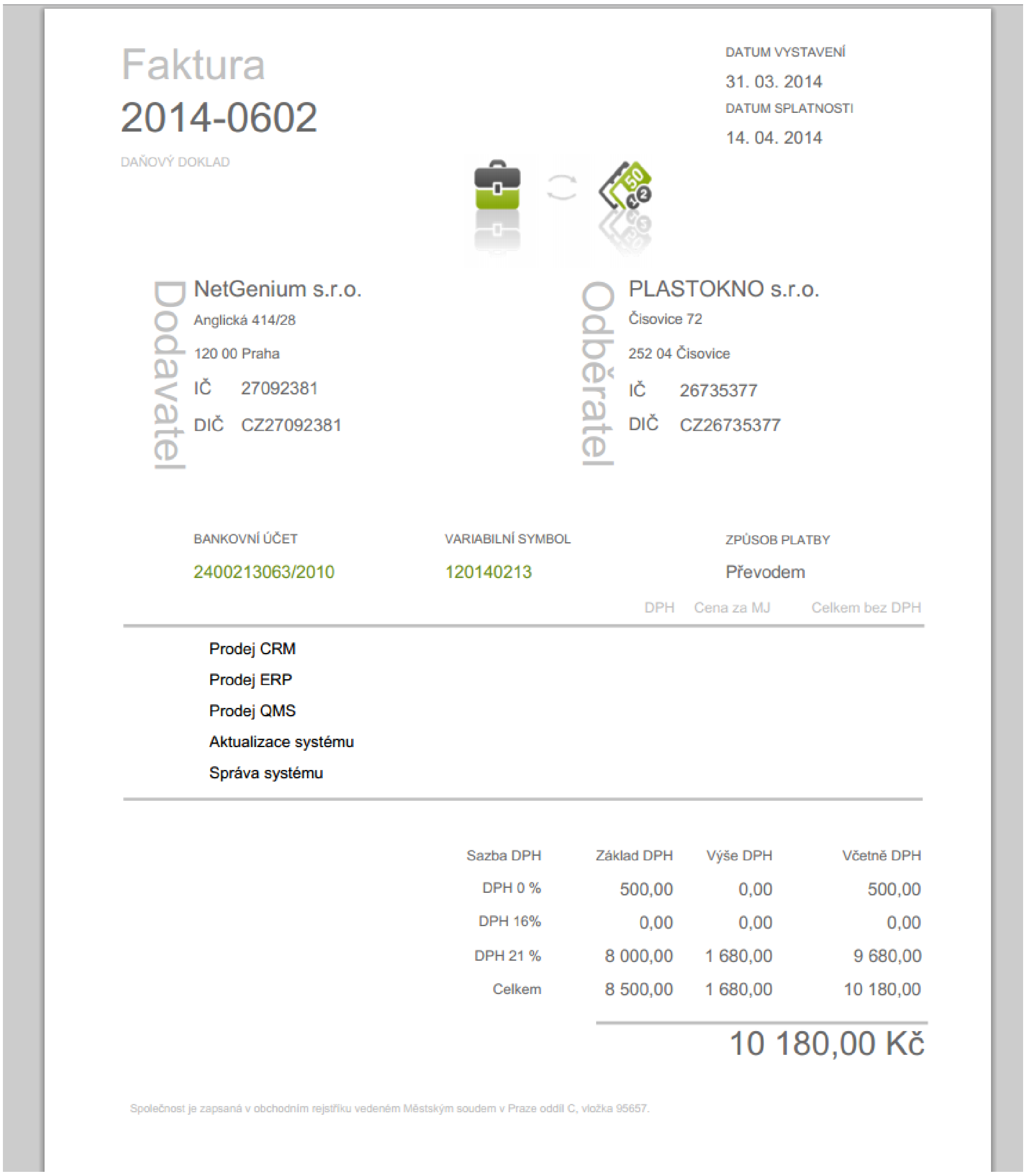
* Invoice number	2022-0001

Invoice items 5 / 5
Item All
Sale CRM
Sale ERP
Sale QMS
System update
System administration

When pressed, a pop-up window will pop up, which will be used to download the "pdf file".


File download - Google Chrome
dev.netgenium.com/martin-sevcik/(S(pkrxs0mka2q0bis0hhw0icbx))/DownloadFile.aspx?XH4Rk3PPaP5J6VMR/jkBOfyf/0NBtH8ePX2...
Invoice.pdf (25,05 kB)
Please wait...
Close
Invoice.pdf
Zobrazit vše

It depends on the browser in which NET Genium is running. For example, in the Mozzila Firefox browser, a "pdf" document is downloaded immediately; in the Google Chrome browser, the generated document opens in a new browser tab. In all cases, however, the generated template should look the same.



Faktura

2014-0602

DAŇOVÝ DOKLAD

DATUM VYSTAVENÍ

31. 03. 2014

DATUM SPLATNOSTI

14. 04. 2014

Dodavatel

NetGenium s.r.o.
Anglická 414/28
120 00 Praha
IČ 27092381
DIČ CZ27092381

Odběratel

PLASTOKNO s.r.o.
Čisovice 72
252 04 Čisovice
IČ 26735377
DIČ CZ26735377

BANKOVNÍ ÚČET	VARIABILNÍ SYMBOL	ZPŮSOB PLATBY
2400213063/2010	120140213	Převodem

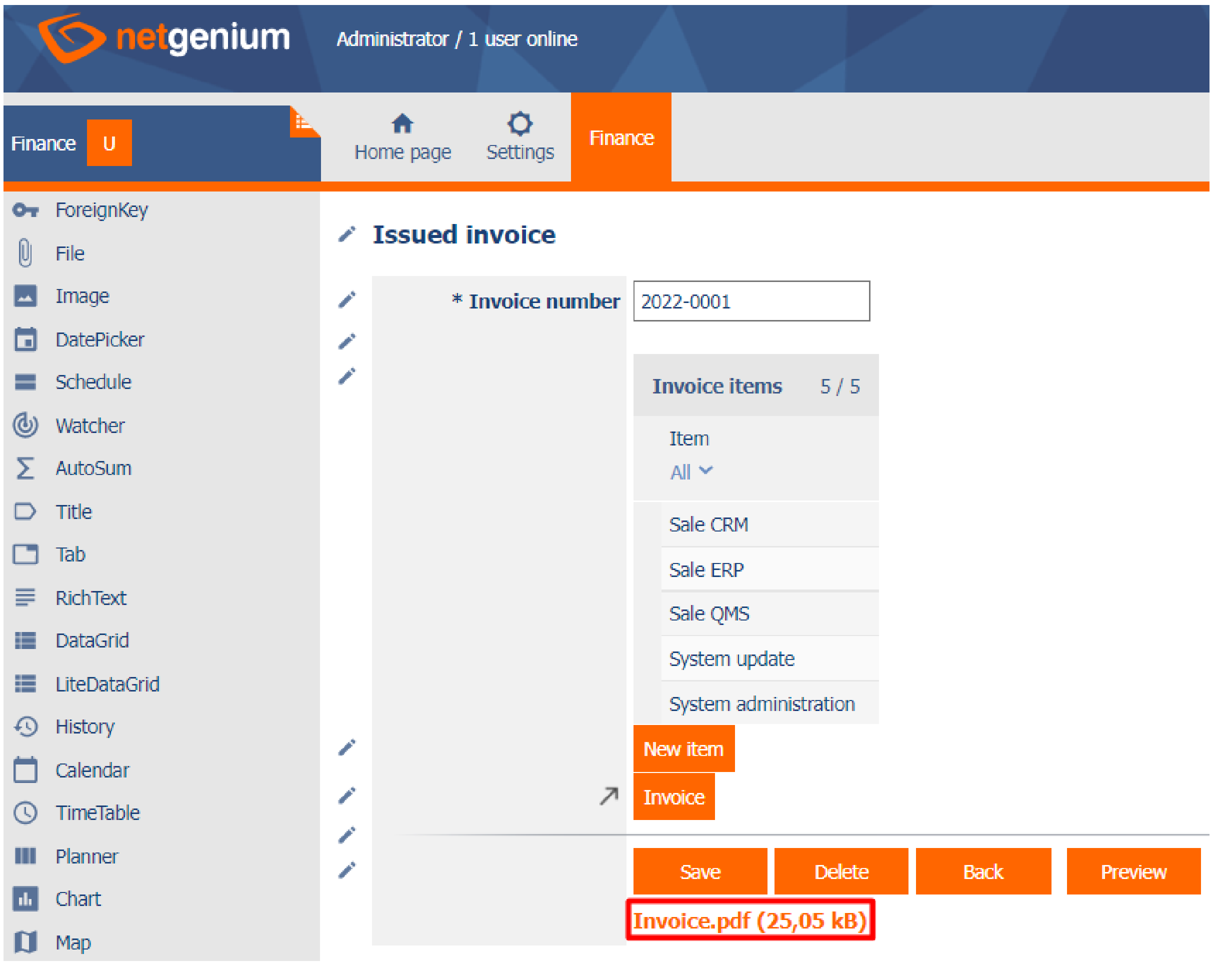
	DPH	Cena za MJ	Celkem bez DPH
Prodej CRM			
Prodej ERP			
Prodej QMS			
Aktualizace systému			
Správa systému			

Sazba DPH	Základ DPH	Výše DPH	Včetně DPH
DPH 0 %	500,00	0,00	500,00
DPH 16%	0,00	0,00	0,00
DPH 21 %	8 000,00	1 680,00	9 680,00
Celkem	8 500,00	1 680,00	10 180,00

10 180,00 Kč

Společnost je zapsaná v obchodním rejstříku vedeném Městským soudem v Praze oddíl C, vložka 95657.

Until the user moves somewhere else in the system, the generated document "Invoice.pdf" will always be ready for download in the edit form of the given record. There is no need to let NET Genium generate the same "pdf" document again.



Issued invoice

* Invoice number: 2022-0001

Invoice items	5 / 5
Item All	
Sale CRM	
Sale ERP	
Sale QMS	
System update	
System administration	

Invoice.pdf (25,05 kB)