

681

" ø - , 2012. 9 : , , "

. . .

, : , ,

, , ó

. ó

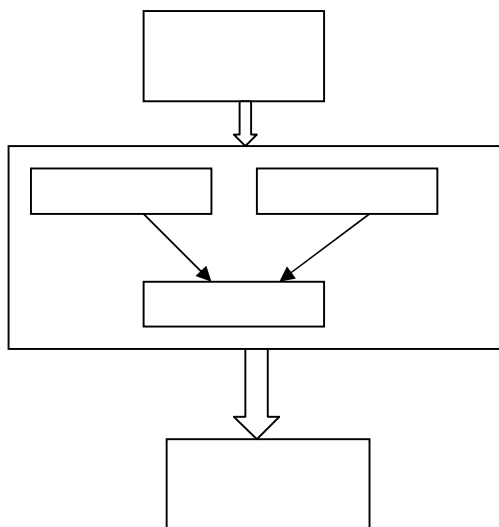
ASP.Net PHP, HTML ó ; SQL

(DDL DML); ;

ø - ;

(.1):

- 1) (,);
- 2) (,);
- 3) (,);



.1

- : ().
- ; ,
- ; ,
- ().
- ;
- ;

```

a = 1;
b = 2;
//x=a+b;
//y=b-a;
),
List<string> program = new List<string>();
string line;
using (StreamReader sr = File.OpenText(filepath))
{
    while (!sr.EndOfStream)
    {
        line= sr.ReadLine();
        if (line.Contains("//"))
        {
            program.Add("int " + line [2] + ";");
            program.Add(line.Substring(2));
        }
        else program.Add(line);
    }
}
Output.PutResult(program, filepath);

```

StreamReader. filepath


```

class MyClass
{
    static void Main()
    {
        Console.WriteLine("Hello!");
        Console.ReadLine();
    }
}

```

XML.

XSLT

```

<?xml-stylesheet type="text/xsl" href="variables.xsl"?>
<VARIABLES>
  <VARIABLE Active="true">
    <VAR>A</VAR>
    <VAL>1</VAL>
  </VARIABLE>
  <VARIABLE Active="true">
    <VAR>B</VAR>
    <VAL>2</VAL>
  </VARIABLE>
  <VARIABLE Active="true">
    <VAR>C</VAR>
    <VAL>3</VAL>
  </VARIABLE>
  <VARIABLE Active="false">
    <VAR>D</VAR>
    <VAL>4</VAL>
  </VARIABLE>
</VARIABLES>

```

D. Active false.
true, D variables.xsl.

XML-

```

<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <xsl:for-each select="VARIABLES/VARIABLE[@ Active='true']">
      const int <xsl:value-of select="VAR"/> =
      <xsl:value-of select="VAL"/>;<BR/>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>

```

VARIABLE, Active,
true, "const int",
(VAR),
(VAL).
const int A = 1;
const int B = 2;
const int C = 3;

XSLT

Visual Studio

T4.

T4.

Visual Basic

C#

```

<#@ template debug="false" language="C#" #>
<#@ output extension=".cs" #>
<# string[] vars = new string [] { "A", "B", "C" }; #>
class MyClass {
<#
  foreach (string variable in vars)
  { #>
    private int <#= variable #> = 0;
  <# } #>
}

```

PHP ASP.Net.

<# #>

```

class MyClass {
  private int A = 0;
  private int B = 0;
  private int C = 0;
}

```

```

List<string> program = new List<string>();
string line;
Regex pattern = new Regex(@"^\s*(?<val>.*)\s*=", RegexOptions.Singleline);
using (StreamReader sr = File.OpenText(filepath))
{
  while (!sr.EndOfStream)
  {
    line = sr.ReadLine();
    foreach (Match m in pattern.Matches(line))
    {
      program.Add("private int " + m.Groups["val"].Value + ";");
    }
    program.Add(Regex.Replace(line, @"^\s*", ""));
  }
}
Output.PutResult(program, filepath);

```

^\s* (?<val>.*) \s*=".

<val>),

```

// a = 1;
// b = 2;
// c = a + b;

```

```

private int a;

```

```
a = 1;
private int b;
b = 2;
private int c;
c = a + b;
```

1. <http://www.intuit.ru/department/se/incodegen/2/>
2. Vogel P Practical Code Generation in .NET : Covering Visual Studio 2005, 2008 and 2010 Boston: Pearson Education, Inc., 2010.
3. Kelly S., Tolvanen J Domain-specific modeling : Enabling Full Code Generation Hoboken, New Jersey: John Wiley & Sons, Inc., 2008.
4. Fertalj K., Brcic M A Source Code Generator Based on UML Specification International journal of computers and communications, Issue 1, Volume 2, 2008.