Consultingwerk
software architecture and development

Regular Expressions for ABL Developers

PUG Challenge 2026 Amsterdam

Sebastian Wandel, Consultingwerk



Sebastian Wandel

Consultingwerk

- Software Developer
- 6+ years with Consultingwerk

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Six Parts, from Syntax to Live Demos



1 What Is Regex?

A tiny language for describing text



2 Regex Syntax

From everyday shorthands to lookarounds



3 Regex in Your IDE

VS Code and Progress Developer Studio



4 Regex in ABL Code

The .NET Regex class via the CLR bridge



5 Live Demos

Finding, refactoring, validating



6 Wrap-up

Take-aways and questions

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.



1 What Is Regex?

A tiny language for describing text

One Pattern Instead of a Thousand IFs

```
Pattern
^\d{4}-\d{2}-\d{2}$
```

```
Input              Result
2026-11-05         match
5.11.2026          no match
2026-11-5          no match
```



A pattern describes a shape

- Not one string, but **every** string of that shape
- Four digits, dash, two digits, dash, two digits
- Find it, check it, replace it, split by it

One declarative line instead of a pile of `substring`, `index` and `entry` calls

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Regex Is Already in Your Toolbox

Search & Replace in the IDE

- VS Code and PDSOE: one toggle away
- One file or the whole workspace

Validation & Parsing in Code

- The .NET `Regex` class via the CLR bridge
- E-mail addresses, VAT numbers, log files

Bulk Refactoring

- Rename packages, move folders, modernize syntax
- Hundreds of files in one operation

ABL has no native regex yet. OPENEDGE-I-217 is "Future consideration". Add your vote.

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Every Tool Brings Its Own Regex Engine

Where	Engine	Scope
VS Code editor (Find widget)	JavaScript (ECMAScript)	Current file
VS Code Search view	ripgrep, PCRE2 when needed	Files in the workspace
PDSOE (Eclipse)	Java `java.util.regex`	File and workspace search
ABL code	.NET `System.Text.RegularExpressions`	At runtime

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.



2 Regex Syntax

From everyday shorthands to lookarounds

Most Characters Just Match Themselves

Literals

- `customer` matches "customer"
- Letters, digits, punctuation
- Case-sensitive unless you switch it off

Metacharacters

- `. ^ $ * + ? ( ) [ ] { } | \`
- Escape them with `\` to match literally
- `end\.` finds "end.", `end.` also finds "ends"

Dot and Anchors

- `.` any character except a line break
- `^` start of the line
- `$` end of the line

In code you escape twice: `\.` for the regex, then `~\.` for the ABL string

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Six Shorthands Do Most of the Work

Token	Matches	ABL example
`\w`	Letter, digit or _	`c`, `i1`, `_tmp`
`\W`	Anything else	space, `.`, `(`, `-`
`\s`	Space, tab, line break	between `define` and `variable`
`\S`	Anything but whitespace	`no-undo.`
`\d`	Digit	`2026`
`\D`	Anything but a digit	`v`, `.`



`\w` has no hyphen

- `\w+` on `cust-name` finds `cust` and `name`
- ABL names need `[\w-]+`

Uppercase negates: `\S` is everything `\s` is not

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Word Boundaries Find Whole Names

```
Class names
Customer
CustomerManager
CustomerManagerFactory
OrderCustomerLookup
LegacyCustomer
```

`\bCustomer\b`

Only the class `Customer`

`\bCustomer\B`

Names that **start** with Customer

`\BCustomer\B`

Customer **inside** a name:
`OrderCustomerLookup`

`\BCustomer\b`

Names that **end** with Customer

`\b` sits between `\w` and `\W`. It matches a position, not a character.

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Quantifiers Say How Many

Token	How many
`*`	zero or more
`+`	one or more
`?`	zero or one (optional)
`{4}`	exactly four
`{2,}`	two or more
`{1,3}`	one to three



Greedy by default, lazy on request

- `<.+>` on `<a><b>` matches `<a><b>`
- `<.+?>` on `<a><b>` matches `<a>`
- Greedy takes as much as it can, lazy as little as it can

`define\s+variable` finds the declaration, however many spaces are in between

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Character Classes Pick Exactly One Character

Class	Matches one...
`[aeiou]`	vowel
`[a-z0-9]`	lowercase letter or digit
`[^"]`	character that is not a quote
`[\w-]`	word character or hyphen
`[.]`	literal dot, no escape needed



Special inside a class

- `-` is a range unless first or last: `[\w-]`
- `^` negates only at the start: `[a^]`
- `]` and `\` need a backslash: `[\]\\]`



Quantify the class

`[\w-]+` matches `cust-credit-limit`

`"[^"]*"` matches a complete string literal (ignoring `~"` escapes)

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Alternation Picks One of Several Words

```
Inconsistently formatted code
define public    variable cName as character no-undo.
DEFINE PROTECTED VARIABLE iCount AS INTEGER.
define   variable lOk as logical no-undo.
```

```
Pattern (case-insensitive)
define\s+
(?:(public|protected|private)\s+)?
variable
```

`a|b`

One of the alternatives

`(?:...)`

Groups without capturing

All three lines match. A plain text search finds none of them reliably.

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Lookarounds Check Without Consuming

Kind	Syntax	Means
Lookahead	`(?=x)`	followed by x
Negative lookahead	`(?!x)`	not followed by x
Lookbehind	`(?<=x)`	preceded by x
Negative lookbehind	`(?<!x)`	not preceded by x

```
Examples
(?<=class\s)(?:[\w-]+\.)*[\w-]+
  -> demo3.EMailHelper

\w+(?=\s*\()
  -> method names before "("

define\s+variable(?!.*no-undo)
  -> variables without no-undo
```

Lookarounds describe a position in the text, not the text you match

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Excluding a Word Needs a Lookahead

```
Wrong: [^...] excludes characters
as\s+[^character]\w*
  -> also misses handle, char

Right: (?!...) excludes a word
as\s+(?!character\b)\w+
  -> integer, handle, logical
```



Look past the next character

- `variable(?!no-undo)` only checks the text right after `variable`
- `variable(?!.*no-undo)` steps over everything in between
- Without the `.*` the check almost always passes

`[^...]` excludes characters, `(?!...)` excludes words

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Named Groups Give Your Captures a Meaning

```
Numbered
([\w-]+)\s*=\s*\1\s+[-+*/]\s+\d+
  -> iCount = iCount + 1
```

```
Named
(?<var>[\w-]+)\s*=\s*\k<var>\s+
  [-+*/]\s+\d+
  -> cust-no = cust-no * 10
```

Group	In pattern	In replace
( )	\1	$1
(?<name>)	\k<name>	${name}



Why names?

They survive a new group added in front

They document what you capture

A backreference matches the same text again, not the same pattern

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Captures Rebuild the Text in Replace

```
Search  define\s+(public|protected)\s+variable\s+([\w-]+)\s+as\s+(\w+)(\s+no-undo)?\s*\.
Input   define public variable cust-name as character no-undo.
```

```
Replace define $1 property $2 as $3$4 get. set.
Result  define public property cust-name as character no-undo get. set.
```

Numbered `$1`

- Works in every tool
- Shifts when you add a group in front

Named `${name}`

- Stays stable, says what it holds
- Not in every tool (VS Code: no)

Numbers are portable, names survive changes to the pattern

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Your One-Slide Cheat Sheet

```
Characters
.          any char
\w \W      word / not
\s \S      space / not
\d \D      digit / not
[abc]      one of
[^abc]     none of
\.         literal dot
```

```
Quantity & Position
*          0 or more
+          1 or more
?          0 or 1
{n,m}      n to m
*? +?      lazy
^ $        line start/end
\A \z      text start/end
\b \B      boundary / not
```

```
Groups & Lookarounds
(x)        capture
(?:x)      group only
(?<n>x)    named
a|b        a or b
\1 \k<n> backreference
(?=x) (?!x)
(?<=x) (?<!x)
```

```
Replace
$1 $2  by number    ${name}  by name    $0  whole match    $$  literal $
```

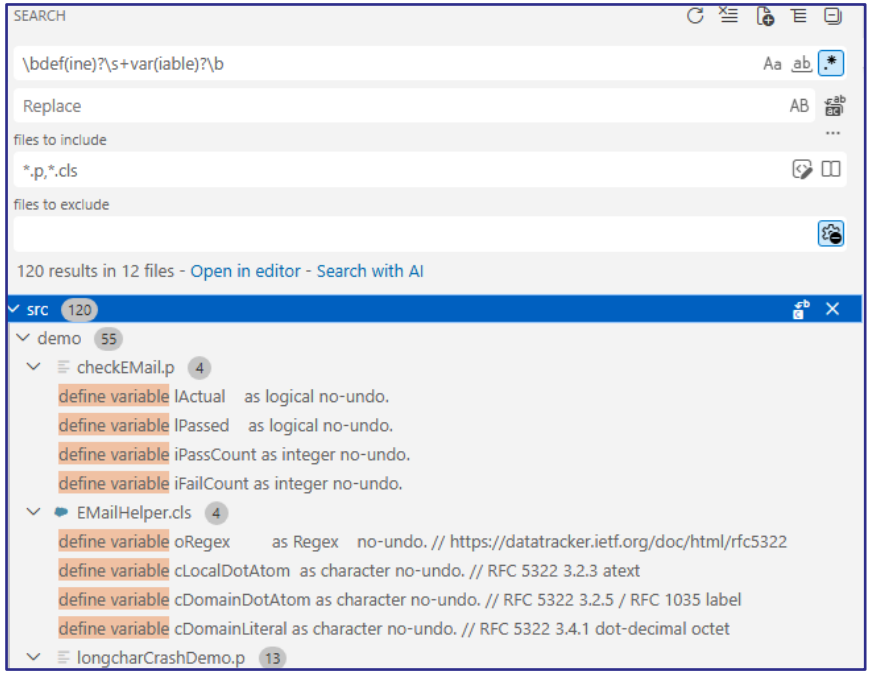
© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

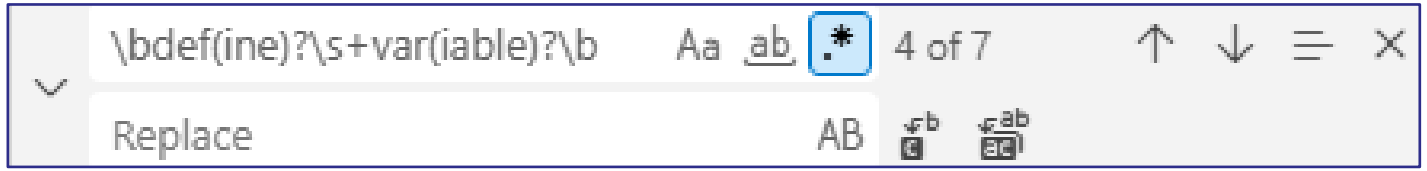


3 Regex in Your IDE

VS Code and Progress Developer Studio for OpenEdge

VS Code: Regex Is One Click Away





In one file

- `Ctrl+F` find
- `Ctrl+H` replace
- `Alt+R` regex

In the workspace

- `Ctrl+Shift+F` find
- `Ctrl+Shift+H` replace
- Globs, diff preview

The editor uses JavaScript regex, the Search view uses ripgrep

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

VS Code Forgets Your Group Names in Replace

```
Search
(?<access>public|protected)\s+variable

Replace (does not work)
${access} property

Replace (works)
$1 property
```

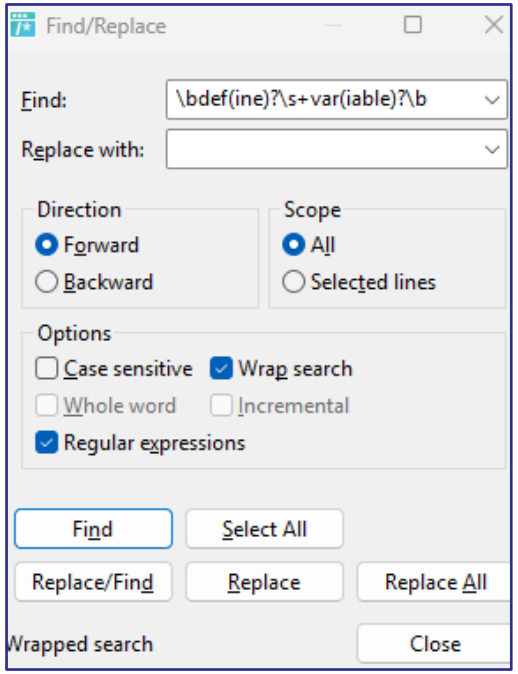
Bonus: change case in replace

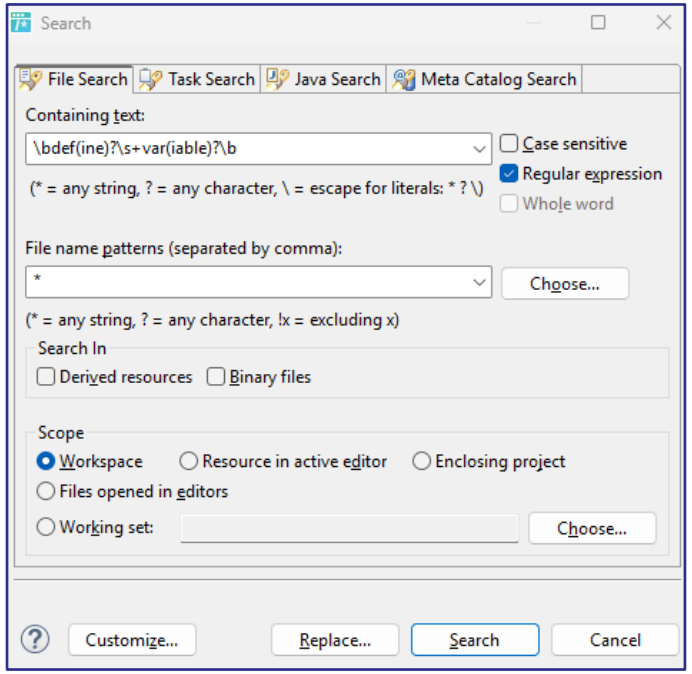
- `\u` / `\l` next character
- `\U` / `\L` until the end
- `\U$1` turns `cust` into `CUST`
- Not available in PDSOE or .NET

Name your groups for readability, count them for the replace string

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

PDSOE Speaks Java Regex





Find/Replace

`Ctrl+F`, check "Regular expressions"

`Ctrl+Space` offers regex syntax

File Search

`Ctrl+H`, scope workspace or project

"Replace..." shows a preview

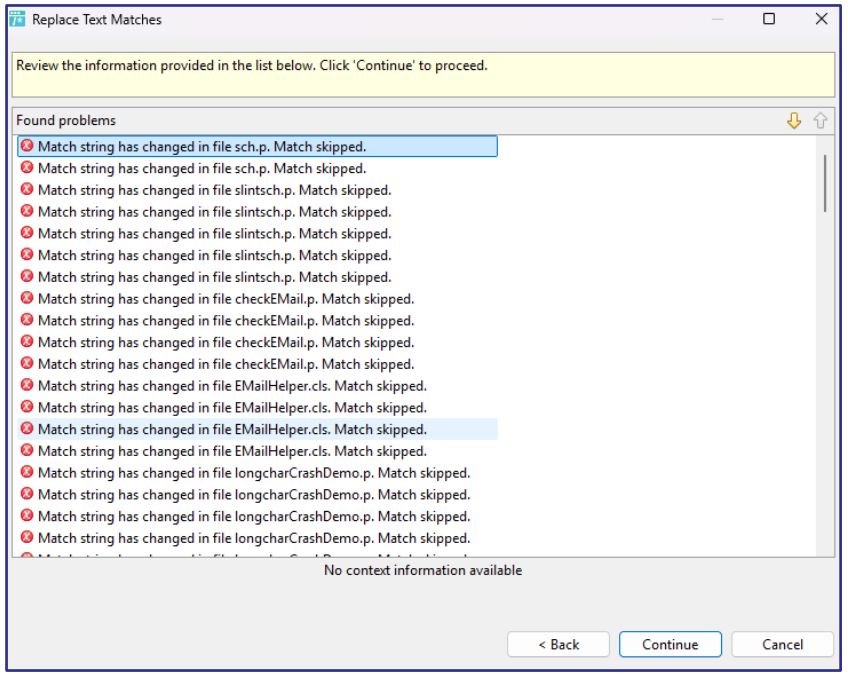
`\R` matches any line break, `$1` inserts a group

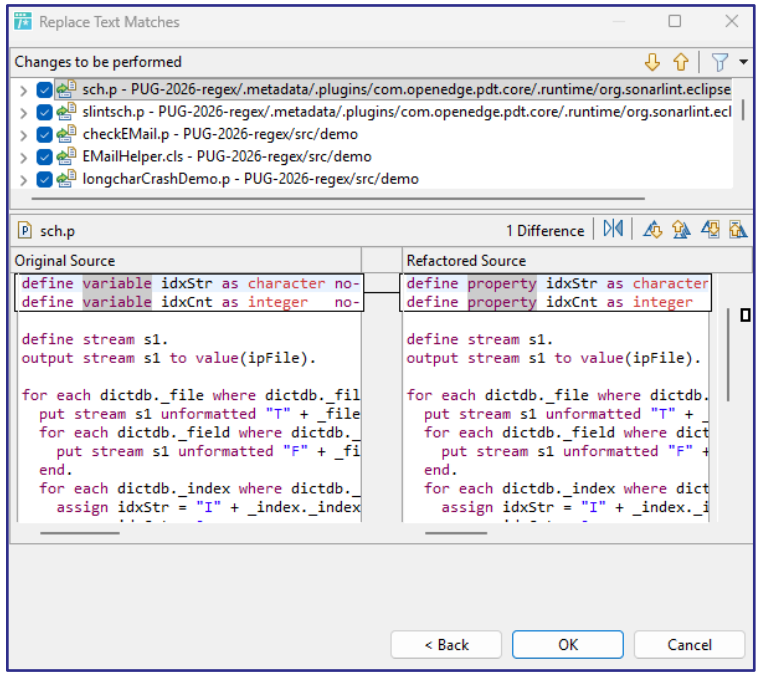
© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

PDSOE Finds with Lookarounds, but Cannot Replace

```
With lookbehind: skipped
(?<=define\s+)variable  ->  property
```

```
Without lookaround: replaced
define\s+variable  ->  define property
```





© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Test in One File, Then Trust the Workspace

	VS Code editor	VS Code Search	PDSOE Find	PDSOE File Search
Lookarounds	yes	yes	search only	search only
Backreferences `\1`	yes	yes	yes	yes
`$1` in replace	yes	yes	yes	yes
`${name}` in replace	no	no	yes	yes
`\U \L` case change	yes	yes	no	no

Same toggle, different engine: try the pattern on one file first

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.



4 Regex in ABL Code

The .NET Regex class via the CLR bridge

One USING Statement Away

```
using System.Text.RegularExpressions.* from assembly.
```



Regex

- The pattern, ready to use
- Match, replace, split



Match, Group, Capture

- The results
- Value, Index, Length



RegexOptions

- Flags that change matching
- Combined with `or`

Linux too, with `-clrnetcore`. More in Peter Judge's ".NET on Linux. Why bother?"

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

ABL Adds Its Own Escaping Layer

Character	Regex	In ABL
Preprocessor braces	`{4}`	`~{4~}`
String delimiter	`"`	`~"`
ABL escape character	`~`	`~~`
Backslash (portable)	`\d`	`~\d`



Read it twice

- ABL strips the `~` at compile time
- Test the unescaped pattern



User input

- Pattern: `Regex:Escape (c)`
- Replacement: `$` becomes `$$`

`\d{4}` becomes `~\d~{4~}` in an ABL string

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Five Methods Cover Almost Everything

Method	Returns	Use it to
`IsMatch`	`logical`	validate input
`Match`	`Match`	get the first hit
`Matches`	`MatchCollection`	get all hits
`Replace`	`character`	substitute hits
`Split`	`character[]`	cut text at hits



Instance or static

- Instance: build once, call often
- Static `Regex:IsMatch` for one-off checks

Every method also takes a start position, `Replace` also a callback

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

A Match Holds Groups, a Group Holds Captures

Regex

- `Match`, `Matches`
- `IsMatch`
- `Replace`, `Split`
- `GetGroupNames()`
- `Options`

Match

- `Success`
- `Value`, `Index`, `Length`
- `Groups`
- `NextMatch()`
- All hits: `MatchCollection`

Group

- `Groups[0]`: the whole match
- `Item[1]`, `Item["name"]`
- `Success`, `Value`
- `Captures`

Capture

- One per repetition
- `Value`, `Index`, `Length`
- The group keeps the last one

`Index` counts from 0: add 1 before you hand it to `substring`

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Walking the Match Object

```
Object model
Match
 +- Groups
     +- Group
         +- Captures

NextMatch() -> next Match
```

```
src/samples/walkMatches.p
oMatch = oRegex:Match (lcContent).
do while oMatch:Success:
    cResult = substitute ("&1&2~n":u, cResult,
        oMatch:Groups:Item["classname":u]:Value).
    oMatch = oMatch:NextMatch ().
end.
```

A group that did not match returns `""`, not `?`: check `:Success`

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

CHARACTER, LONGCHAR or a Whole File

CHARACTER

- Converted to `System.String` on every call
- Cheap: at most about 32 KB
- `inputCharacter.p`

LONGCHAR

- Same conversion, on every call
- Costly for very large text
- `inputLongchar.p`

A whole file

- `ReadAllText` into a `System.Object`
- `Match` called via reflection
- `inputFile.p`

For big inputs, let .NET do the reading and keep the text on the .NET side

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

RegexOptions Change the Rules

Option	Inline	Effect
`IgnoreCase`	`(?i)`	Case-insensitive matching
`Multiline`	`(?m)`	`^` and `$` match at every line, not just at the start and end
`Singleline`	`(?s)`	`.` also matches line breaks
`ExplicitCapture`	`(?n)`	Only named groups capture, `( )` acts like `(?:)`
`IgnorePatternWhitespace`	**`(?x)`**	**Whitespace in the pattern is ignored: lay it out over lines**
`Compiled`		Pattern becomes .NET IL code: slower to build, faster to run

Combine them with `or`: `RegexOptions:Compiled or RegexOptions:IgnoreCase`

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Multiline Moves the Anchors, Singleline Moves the Dot

	Default	With `Multiline`
`^`	start of the text	start of every line
`$`	end, or before a final `\n`	end of every line
`\A`	start of the text	unchanged
`\Z`	end, or before a final `\n`	unchanged
`\z`	the very end	unchanged

```
Input first\nsecond\nthird

Pattern         none  m     s     m+s
^second$        no    yes   no    yes
first.second    no    no    yes   yes
^second.third   no    no    no    yes
\Asecond        no    no    no    no
```

Validate with `\A...\z`, and remember: `$` stops before `\n`, not before `\r\n`

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Compile Once, Reuse Forever

```
define private static variable oRegex
    as Regex no-undo.

define static property VatRegex as Regex
    no-undo
get():
    if not valid-object(oRegex) then
        oRegex = new Regex("^DE~\d~{9~}$":u,
                           RegexOptions:Compiled).
    return oRegex.
end get.
```



Compiled

Costs time once

Pays off on every later call



Never in a loop

Build the `Regex` once

Keep it in a static property

The same shape powers `VatHelper` and `EMailHelper` in the demos

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Which One Would You Rather Maintain?

Regex

```
^\s*def(?:ine)?\s+(?:(?:publi
c|protected|private)\s+)?var(
?:iable)?\s+([\w-]+)\s+as\s+(
[\w.]+)(?:\s+no-undo)?\s*\.\s*
$
```

ABL

```
c=trim(replace(p,"~t"," ")).
do while index(c,"  ")>0: c=replace(c,"  "," "). end.
if substr(c,length(c))<>"." then return.
c=right-trim(substr(c,1,length(c) - 1)).
if lookup(entry(1,c," "),"def,define")=0 then return.
i=2. if lookup(entry(i,c," "),"public,protected,private")
>0 then i=i + 1. if lookup(entry(i,c," "),"var,variable")
=0 or entry(i + 2,c," ")<>"as" then return.
n=entry(i + 1,c," "). t=entry(i + 3,c," ").
```

Unreadable is a writing style, not a property of regex

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Treat a Regex Like Source Code

```
Regex with IgnorePatternWhitespace
^ \s*
def(?:ine)? \s+     (?# def|define )
(?: (?:public
     | protected
     | private
    ) \s+
)?                  (?# access )
var(?:iable)? \s+   (?# var|variable )
(?<name> [\w-]+ )   (?# may contain - )
\s+ as \s+
(?<type> [\w.]+ )   (?# qualified )
(?: \s+ no-undo )?
\s* \. \s* $
```

```
ABL, formatted
/* one space between all words */
cStmt = trim(replace(pcStmt, "~t", " ")).
do while index(cStmt, "  ") > 0:
    cStmt = replace(cStmt, "  ", " ").
end.
/* must end with a period */
if substring(cStmt, length(cStmt)) <> "."
    then return.
cStmt = right-trim(cStmt, " .").
if lookup(entry(1, cStmt, " "),
          "def,define") = 0 then return.
/* optional access modifier */
iPos = 2 + (if lookup(entry(2, cStmt, " "),
  "public,protected,private") > 0
  then 1 else 0).
...
```

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

.NET Lets Branches Share One Group Name

```
src/samples/VatHelper.cls (shortened)
^(?:
     (?<country>AT)(?<number>U\d{7}(?<check>\d))
   | (?<country>DE)(?<number>\d{8}(?<check>\d))
   | (?<country>DK)(?<number>\d{8})
   | (?<country>NL)(?<number>\d{9}B\d{2})
)$
```



One name, many branches

- Every branch fills `country`, `number`
- `check` where the format has one
- No `if` per country
- Java: no duplicate names, PCRE: `(?J)`

A .NET feature most other engines forbid

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.



5 Live Demos

Finding, refactoring, validating

Live Demo 1: Finding Code in PDSOE and VS Code



Before

- `shop`: five classes named like `Customer`
- Mixed case, `def var`, extra spaces

```
» Patterns:
\bCustomer\B\w*Manager\b

def(?:ine)?\s+(?:\w+\s+)?
  var(?:iable)?\s+\w*Count\b

\bnew\s+\w*Customer\w*\s*\(
```

You'll see

- Same hits in both IDEs
- Nothing a text search could separate

Plain text search finds strings. Regex finds code.

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Live Demo 2: Move a Folder, Fix Every Reference



Before

- The folder `app/legacy` is dissolved
- Its files move one level up
- References in code, XML and JSON

```
» Search & Replace:
\bapp([\./\\]{1,2})legacy\1

app$1

files to include:
*.cls,*.p,*.i,*.xml,*.json
```

You'll see

- One preview with every reference
- One click to fix them all

., /, \ and \\ handled by one pattern and one backreference

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Valid or Not? You Decide

```
Raise your hand if you think the address is valid

user@localhost
"john doe"@example.com
-user@example.com
user@[999.168.0.1]
```

The answers come from the RFCs, not from gut feeling

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Live Demo 3: Validating E-Mail Addresses





Before

- 14 addresses in `email-test-cases.xml`
- Votes from the audience

```
» Run:
src/demo3/checkEMail.p

EMailHelper:
  IsValidEMailAddress(c)
```



You'll see

- Pass/fail per address
- A formatted, commented RFC 5322 regex

RFC 5322 has more surprises than most validation code has test cases

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Live Demo 4: Turning Variables into Properties



Before

- `classic`: four old-style classes
- Public variables, old package name

```
» Run:
src/demo4/refactor.p

classic  ->  modern
```

You'll see

- Properties instead of variables
- Renamed package, += assignments

Five regex objects, hundreds of edits, one run

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.



6 Wrap-up

Five Things to Take Home

Engines differ

Same toggle, different features

One file first

Then trust the workspace

Build once

Cache `Regex` objects in a static property

Format it

`IgnorePatternWhitespace` and `(?# )` comments

Rare in code, daily in search

Practise in the IDE, even just `\b` and `\s*`, so it is not new when code needs it

Regex is code: format it, test it, review it

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Slides and Samples on GitHub



Everything from today

Slides, demo classes and the e-mail validation

https://github.com/consultingwerk/ABL-RegEx-Samples

Keep practicing

regex101.com

Idea: OPENEDGE-I-217

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Questions



© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.


Consultingwerk
software architecture and development