Sonstige Themen

Ärzte

```
Klaus Weinert
Facharzt für Kinder- und Jugendmedizin in Köln
Görlinger-Zentrum 5 - 7
50829 Köln
Tel +49 221 508887
Fax +49 221 80064591
```

```
Praxis Gerald S. Langner
Facharzt für Kinder- und Jugendpsychiatrie und Psychotherapie
=> Fr. Dr. Paul
Vogelsanger Str. 106-108
50823 Köln
Tel +49 221 5708330
Fax +49 221 57083366
```

```
Gemeinschaftspraxis Bocklemünd
Görlinger-Zentrum 5 - 7
50829 Köln
Tel.: +49 221 501349
```

Öffnungszeiten (Nachmittags nur mit Termin)		
Montag	08:00 – 11:00 Uhr	15:30 – 18:00 Uhr
Dienstag	08:00 – 11:00 Uhr	15:30 – 18:00 Uhr
Mittwoch	08:00 – 11:00 Uhr	
Donnerstag	08:00 – 11:00 Uhr	15:30 – 18:00 Uhr
Freitag	08:00 – 11:00 Uhr	

```
Hausarzt Dr. med. Alexander Lang
Grevenbroicher Str. 25
50829 Köln
Tel.: +49 221 508111
```

Öffnungszeiten		
Montag	09:30 – 12:00 Uhr	16:00 – 18:00 Uhr
Dienstag	09:30 – 12:00 Uhr	16:00 – 18:00 Uhr
Mittwoch	09:30 – 12:00 Uhr	
Donnerstag	09:30 – 12:00 Uhr	16:00 – 18:00 Uhr
Freitag	09:30 – 12:00 Uhr	16:00 – 18:00 Uhr

Syntaxhighlighting

```
@media (max-width: 664px) {
    .wp-block-group:not(.alignfull):not(.alignwide) > .wp-block-
group__inner-container > * {
        width: 302px;
    }
}
```

SELECT CustomerId – , Predecessor_Id

```
   , CustomerNumber
   , VendorNumber
   , DisplayName
   , ShortName
   , ZipCode
   , City
   , OtherVendorNumber
   , OtherVendorName
   , CASE WHEN IsActive=1 THEN 'Active' ELSE 'Inactive' END AS IsActive
   , CASE WHEN IsReadOnly=1 THEN 'ReadOnly' ELSE 'Writable' END AS
IsReadOnly
   , CASE WHEN HasFineCutDiscount=1 THEN 'FCD' ELSE 'No FCD' END AS
HasFineCutDiscount
   , (SELECT DisplayName FROM dbo.masUser WHERE UserId = KeyAccountManager)
AS KeyAccountManager
   , (SELECT DisplayText FROM dbo.masCustomerLevel WHERE CustomerLevelId =
PrimaryCustomerLevel) AS PrimaryCustomerLevel
   , (SELECT DisplayText FROM dbo.masCustomerLevel WHERE CustomerLevelId =
SecondaryCustomerLevel) AS SecondaryCustomerLevel
FROM dbo.masCustomer
```

WHERE CustomerId IN (1, 8, 18, 99, 102, 145, 169, 227, 246, 286, 360)

Visual Basic

```
Case "PROSKURNIN", "CAMES", "INTERTABAK", "MACK", "SZZ", "BRUECK"
    Filename = CreatingFilename(pRow("LocalFilePath").ToString,
pRow("LocalFilePrefix").ToString, strTakeorderNumber, "CSV")
    Dim OutputFile As CSVFile = New CSVFile(Filename)
    If Organisation = "CAMES" Then
        OutputFile.ExportMode = "Plain"
        OutputFile.CustomerFields = {"KdNr beim Spediteur", "Name",
"Bestellnummer", "Bestelldatum", "Bestellinfo", "Position", "Artikelnummer",
            "Menge", "Mengeneinheit", "GTIN", "Preislistenname", "Packungen
pro Karton", "Inhalt", "Preis", "Preisliste"}
        OutputFile.Export(pRow, strTakeorderNumber, OriginOfOrder, relation)
    ElseIf Organisation = "SZZ" Then
        OutputFile.ExportSZZ(pRow, strTakeorderNumber, OriginOfOrder,
relation)
```

```
    ElseIf Organisation = "BRUECK" Then
        OutputFile.ExportMode = "Excel"
        OutputFile.CustomerFields = {"Name", "Strasse", "PLZ", "Ort",
"Bestellnummer", "Bestelldatum", "Bestellinfo", "Position", "Artikelnummer",
            "Menge", "Mengeneinheit", "Preislistenname", "Preis",
"Preisliste", "Packungen pro Karton", "Inhalt"}
        OutputFile.Export(pRow, strTakeorderNumber, OriginOfOrder, relation)
    Else   ' PROSKURNIN, MACK und INTERTABAK
        OutputFile.ExportMode = "Excel"
        OutputFile.Export(pRow, strTakeorderNumber, OriginOfOrder, relation)
    End If
```

C#

```
if (posibleCustomerIds.Count < 1)
{
    if (cust.DisplayName.Equals("TGV, Sindelfingen",
StringComparison.OrdinalIgnoreCase))
    {
        posibleCustomerIds = await reader.CustomerIdGet(new
FindCustomer("TGV Süd", "1003060", true));
    }
    else if (cust.DisplayName.Equals("Fütterer, St. Leon Rot",
StringComparison.OrdinalIgnoreCase))
    {
        posibleCustomerIds = await reader.CustomerIdGet(new
FindCustomer("Fütterer, Tabak", cust.CustomerNumber, true));
    } ...
}
```

Python

```
class ShikakuSolver:
    def __init__(self, dimension):
        self.width = dimension[0]
        self.height = dimension[1]
        self.rows = 0
        self.values = defaultdict(int)
        self.grid = defaultdict(list)
        self.solution = defaultdict(dict)
        self.solutions = list()
        self.requirements = list()
        self.actions = defaultdict(list)
        self.names = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
        self.id = 0
```

```
    def add_row(self, row):
        for i, value in enumerate(row):
            self.grid[self.rows].append(value)
            self.requirements.append(('cell covered', self.rows, i))
            if value > 0:
                self.values[(self.rows, i)] = value
                self.requirements.append(('value covered', self.rows, i))
        self.rows += 1
        if self.rows == self.height:
            self.set_shapes()

    def set_shapes(self):
        for p in self.values.keys():
            r = p[0]
            c = p[1]
            n = self.values[(r, c)]
            i = 1
            shapes = set()
            while i <= n:
                if n % i == 0 and i <= self.height and n // i <= self.width:
                    shapes.add((i, n // i))
                i += 1
            for shape in shapes:
                h = shape[0]
                w = shape[1]
                self.place_shape(r, c, w, h)

    def place_shape(self, rn, cn, w, h):
        r0 = rn - h if rn - h > 0 else 0
        c0 = cn - w if cn - w > 0 else 0
        for r in range(r0, rn + 1):
            for c in range(c0, cn + 1):
                cells = self.check_rectangle(r, c, w, h, rn, cn)
                if r + h <= self.height and c + w <= self.width and cells !=
None:
                    self.actions[('set rectangle', r, c, w, h)] = [('value
covered', rn, cn), *cells]

    def check_rectangle(self, r0, c0, w, h, rn, cn):
        value_included = False
        cells = list()
        for r in range(r0, r0 + h):
            for c in range(c0, c0 + w):
                cells.append(('cell covered', r, c))
                if r == rn and c == cn:
                    value_included = True
                elif (r, c) in self.values.keys():
                    return None
        if value_included:
            return cells
        return None
```

```
    def get_solutions(self):
        solver = AlgorithmXSolver(self.requirements, self.actions)
        for solution in solver.solve():
            self.solution.clear()
            self.id = 0
            for i, action in enumerate(sorted(solution)):
                self.set_rectangle(action)
                self.id += 1
            self.solutions.append(self.solution)
            #break
        print(solver.solution_count)
        return self.solutions

    def set_rectangle(self, action):
        _, row, col, width, height = action
        value = self.names[self.id]
        #print(self.names[i], row, col, width, height)
        for r in range(row, row + height):
            for c in range(col, col + width):
                #print("({}|{})={}".format(r, c, value))
                self.solution[r][c] = value

    def print_grid(self):
        output = "--" * self.width + "-"
        for r in range(self.height):
            output += "\n"
            for c in range(self.width):
                output += "{:2d}".format(self.grid[r][c])
        print(output)
        print("--" * self.width + "-")

# Test the first simple grids:
for riddle in riddles:
    solver = ShikakuSolver(riddle[0])
    for i in range(riddle[0][1]):
        solver.add_row(riddle[i + 1])
    solver.print_grid()
    #print(solver.actions)
    solutions = solver.get_solutions().sort()
    solution = solutions[0]
    for r in range(riddle[0][1]):
        row = ""
        for c in range(riddle[0][0]):
            row += solution[r][c]
        print(row)
```

XML

```
<?xml version="1.0" encoding="iso-8859-15"?>
```

```
<Auftrag>
  <Kopf>
    <Order_Num>0000304362</Order_Num>
    <Order_Dat>20201211</Order_Dat>
    <Liefer_Datum_Wunsch />
    <JTI_GLN_Num>4032800000009</JTI_GLN_Num>
    <GH_GLN_Num>4399901095458</GH_GLN_Num>
    <Name1_WE>Schuller, Margot</Name1_WE>
    <Name2_WE>Lädle Tee &amp;amp; Geschenke</Name2_WE>
    <Strasse_WE>Flamingoweg 1</Strasse_WE>
    <Ort_WE>Stuttgart</Ort_WE>
    <PLZ_WE>70378</PLZ_WE>
    <KdNr_WE_beim_Spediteur>10109</KdNr_WE_beim_Spediteur>
    <KdNr_WE_beim_Lieferant>0000802831</KdNr_WE_beim_Lieferant>
    <Order_Info>ä=ae; Ä=Ae, ö=oe; Ö=Oe; ü=ue; Ü=Ue; ß=sz</Order_Info>
  </Kopf>
  <Positionen>
    <Position>
      <ID>1</ID>
      <GTIN>4032800066821</GTIN>
      <JTI_ArtNr>PR_42150</JTI_ArtNr>
      <JTI_ArtText>WINSTON RED LONGS BP XXL</JTI_ArtText>
      <JTI_PricelistName>WINSTON RED LONGS BP XXL</JTI_PricelistName>
      <PacksPerCarton>8</PacksPerCarton>
      <Content>29</Content>
      <Price>8.0000</Price>
      <PriceList>Z079E</PriceList>
      <Menge>1</Menge>
      <MengenArt>CAR</MengenArt>
      <Position_Info />
    </Position>
    <Position>
      <ID>2</ID>
      <GTIN>4032800054736</GTIN>
      <JTI_ArtNr>PR_42135</JTI_ArtNr>
      <JTI_ArtText>WINSTON VOLUME RED GIANT BOX</JTI_ArtText>
      <JTI_PricelistName>WINSTON VOLUME RED GIANT BOX</JTI_PricelistName>
      <PacksPerCarton>1</PacksPerCarton>
      <Content>280</Content>
      <Price>49.9500</Price>
      <PriceList>F037E</PriceList>
      <Menge>1</Menge>
      <MengenArt>PAC</MengenArt>
      <Position_Info />
    </Position>
  </Positionen>
</Auftrag>
```

SMTP

```
To: "G=MG1;S=MGATE;CN=MG1 MGATE;O=TESTAG;P=MGATE;A=VIAT;C=DE"
<49603@viaT.de>
From: "G=ipm;S=tester;O=testag;A=viaT;C=de" <49637@viaT.de>
Message-ID: 614 07/11/13
X-MPDUID: 8B0663A011DCEC4417009682
Date: 13 Nov 2010 13:10:22 +0100
Subject: Test mit 3 Bodyparts
Disposition-Notification-To: "G=ipm;S=tester;O=testag;A=viaT;C=de"
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="MG_=_CA610D0211DC91E900007CAD_=_MG"
--MG_=_CA610D0211DC91E900007CAD_=_MG
Content-Type: text/plain
Content-Transfer-Encoding: 8bit
Test äöüÄÖÜß
--MG_=_CA610D0211DC91E900007CAD_=_MG
Content-Type: application/octet-stream
Content-Disposition: attachment; filename="4d654d1d.zip
```