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;
    }
}
```

SQL

```
SELECT mass.DisplayText Assignment, mc.CustomerId, mc.ShortName,
COUNT(mc.CustomerId) Counter
  FROM masCustomerAccumulation mca
       INNER JOIN masCustomer mc ON mc.CustomerId = mca.Customer_Id
       INNER JOIN masAccumulation macc ON macc.AccumulationId =
mca.Accumulation_Id
       INNER JOIN masAssignment mass ON mass.AssignmentId =
macc.Assignment_Id
 WHERE mass.DisplayText IN ('Food', 'Gas')
 GROUP BY mass.DisplayText, mc.CustomerId, mc.ShortName
 ORDER BY Assignment, Counter DESC, ShortName
```

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

```
import sys
import math
from collections import defaultdict
from AlgorithmX import AlgorithmXSolver

class LatinSquare:
    possible_values = list()
    multiple_solutions = bool
    empty_value = str
    size = int
    box_size = int
    grid = defaultdict(list)

    def __init__(self, all_possible_values, multiple_solutions_possible =
False):
        self.empty_value = all_possible_values[0]
        self.possible_values = all_possible_values[1:]
        self.multiple_solutions = multiple_solutions_possible
        self.size = len(self.possible_values)
        #print("Empty={}, Values={}, Size={}".format(self.empty_value,
self.possible_values, self.size), file=sys.stderr, flush=True)
```

```
    def add_row(self, new_row):
        i = len(self.grid) // self.size
        row = list(new_row)
        for j in range(self.size):
            if row[j] == self.empty_value:
                self.grid[(i, j)] = self.possible_values
            else:
                self.grid[(i, j)] = [row[j]]

    def get_requirements(self):
        return [('cell covered', row, col) for row in range(self.size) for
col in range(self.size)] + \
               [('value in row', row, val) for row in range(self.size) for
val in self.possible_values] + \
               [('value in col', col, val) for col in range(self.size) for
val in self.possible_values]

    def get_actions(self):
        actions = dict()
        for row in range(self.size):
            for col in range(self.size):
                for val in self.grid[(row, col)]:
                    action = ('place value', row, col, val)
                    actions[action] = [('cell covered', row, col),
                                       ('value in row', row, val),
                                       ('value in col', col, val)]
        return actions

    def get_solution(self):
        solver = AlgorithmXSolver(self.get_requirements(),
self.get_actions())
        result = defaultdict(dict)
        for solution in solver.solve():
            for action in solution:
                _, row, col, val = action
                result[row][col] = val
            if not self.multiple_solutions:
                break
        return result

    def get_solution_count(self):
        #print("Grid={}".format(self.grid), file=sys.stderr, flush=True)
        solver = AlgorithmXSolver(self.get_requirements(),
self.get_actions())
        for solution in solver.solve():
            pass
        return solver.solution_count

    def print_solution(self):
        result = self.get_solution()
        for row in range(self.size):
```

```
            print("".join(result[row][col] for col in range(self.size)))

# Name all values, including the first one, that marks an empty cell
#all_possible_values = "0123456789"
all_possible_values = list("0123456789")
n = 4
square = LatinSquare(all_possible_values[:n-9], True)

#for i in range(len(all_possible_values) - 1):
#    sudoku_solver.add_row(input())
square.add_row("0000")
square.add_row("0000")
square.add_row("0000")
square.add_row("0000")

print("The number of possible solutions is
{}".format(square.get_solution_count()))
```

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
```